精简版(2014.4.7)

编程狂人 第十九期

推酷出品

weibo：http://weibo.com/tuicool2012

Email: mm@tuicool.com

web：www.tuicool.com

注：本期为精简版完整版请点击：
http://www.tuicool.com/mags/534212b3d91b1

也谈基于NodeJS的全栈式开发（基于NodeJS的前端后端分离）

随着不同终端(pad/mobile/pc)的兴起，对开发人员的要求越来越高，纯浏览器端的响应式已经不能满足用户体验的高要求，我们往往 需要针对不同的终端开发定制的版本。为了提升开发效率，前后端分离的需求越来越被重视，后端负责业务/数据接口，前端负责展现/交互逻辑，同一份数据接 口，我们可以定制开发多个版本。
这个话题最近被讨论得比较多，阿里有些BU也在进行一些尝试。讨论了很久之后，我们团队决定探索一套基于NodeJS的前后端分离方案，过程中有一些不断变化的认识以及思考，记录在这里，也希望看到的同学参与讨论，帮我们完善。

一、什么是前后端分离?

最开始组内讨论的过程中我发现，每个人对前后端分离的理解不一样，为了保证能在同一个频道讨论，先就什么是”前后端分离”达成一致。

大家一致认同的前后端分离的例子就是SPA(Single-page application)，所有用到的展现数据都是后端通过异步接口(AJAX/JSONP)的方式提供的，前端只管展现。
从某种意义上来说，SPA确实做到了前后端分离，但这种方式存在两个问题:

- WEB服务中，SPA类占的比例很少。很多场景下还有同步/
同步+异步混合的模式，SPA不能作为一种通用的解决方案。
- 现阶段的SPA开发模式，接口通常是按照展现逻辑来提供的，有时候为了提高效率，后端会帮我们处理一些展现逻辑，这就意味着后端还是涉足了view层的工作，不是真正的前后端分离。

SPA式的前后端分离，是从物理层做区分。（认为只要是客户端的就是前端，服务器端的就是后端）这种分法已经无法满足我们前后端分离的需求，我们认为从职责上划分才能满足目前我们的使用场景:

- 前端：负责View和Controller层
- 后端：只负责Model层，业务处理/数据等。

为什么去做这种职责的划分，后面会继续探讨。

二、为什么要前后端分离?

关于这个问题，玉伯的文章Web研发模式演变中解释得非常全面，这里我根据自己的理解概括一下:

2.1 现有开发模式的适用场景

玉伯提到的几种开发模式，各有各的适用场景，没有哪一种完全取代另外一种。

- 比如后端为主的MVC，做一些同步展现的业务效率很高，但是遇到同步异步结合的页面，与后端开发沟通起来就会比较麻烦。

- Ajax为主SPA型开发模式，比较适合开发app类型的场景，但是只适合做app，因为SEO等问题不好解决，对于很多类型的系统，这种开发方式也过重。

2.2 前后端职责不清

在业务逻辑复杂的系统里，我们最怕维护前后端通吃那种人写的代码，因为没有约束，M-V-C每一层都可能出现别的层的代码，日积月累，完全没有维护性可言。
虽然前后端分离没办法完全解决这种问题，但是可以大大缓解。因为从物理层次上保证了你不可能这么做。

2.3 开发效率问题

淘宝的web基本上都是基于M-VC框架webx，架构决定了前端只能依赖后端。
所以我们的开发模式依然是，前端写好静态demo，后端翻译成vm模版，这种模式的问题就不说了，被吐槽了很久。
直接基于后端的项目开发也很痛苦，一是后端开发环境很重，配置安装使用都很麻烦，而且慢。为了解决这个问题，我们发明了各种工具，比如vmarket，但是前端还是要写vm，而且依赖后端数据，效率依然不高。
另外，后端也没法摆脱对展现的强关注，从而专心于业务逻辑层的开发。
套页面，引脚本就不说了，还要经常让他们改个时间戳，搞个vmcommon，他们也非常痛苦。

2.4 对前端发挥的局限

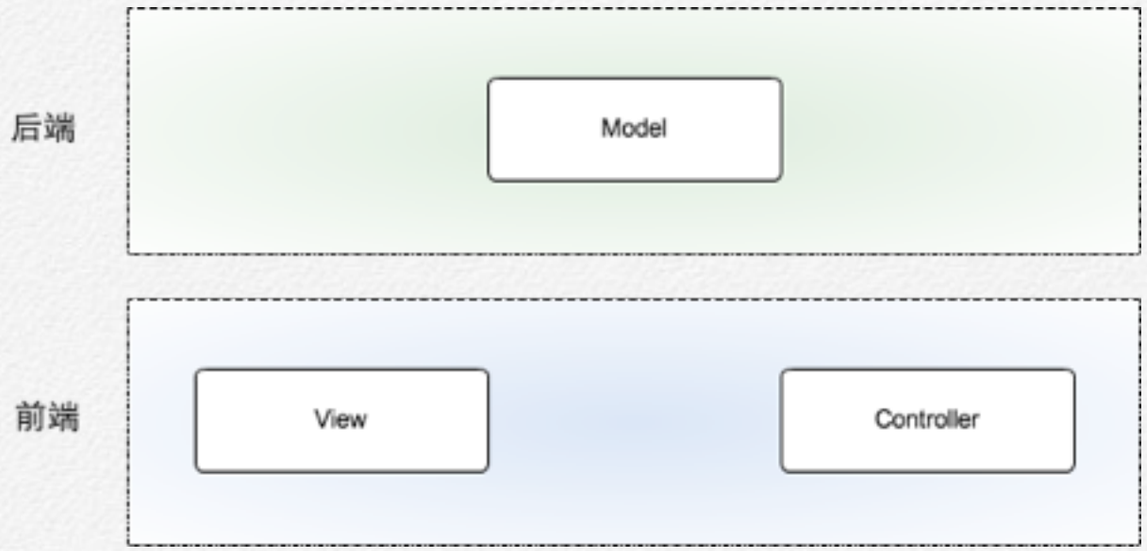
例如，性能优化如果只在前端做空间非常有限，于是我们经常需要后端合作才能碰撞出火花。但是后端由于框架限制，有时候很难做，比如bigpipe，长链接之类的。

为了解决以上提到的一些问题，我们进行了很多尝试，开发了各种工具，但始终没有太多起色，主要是因为我们只能在后端给我们划分的那一小块空间去发挥。只有真正做到前后端分离，把选择权交还给前端，我们才能彻底解决以上问题。

三、怎么做前后端分离?

怎么做前后端分离，其实第一节中已经有了答案:

- 前端：负责View和Controller层
- 后端：只负责Model层，业务处理/数据等。

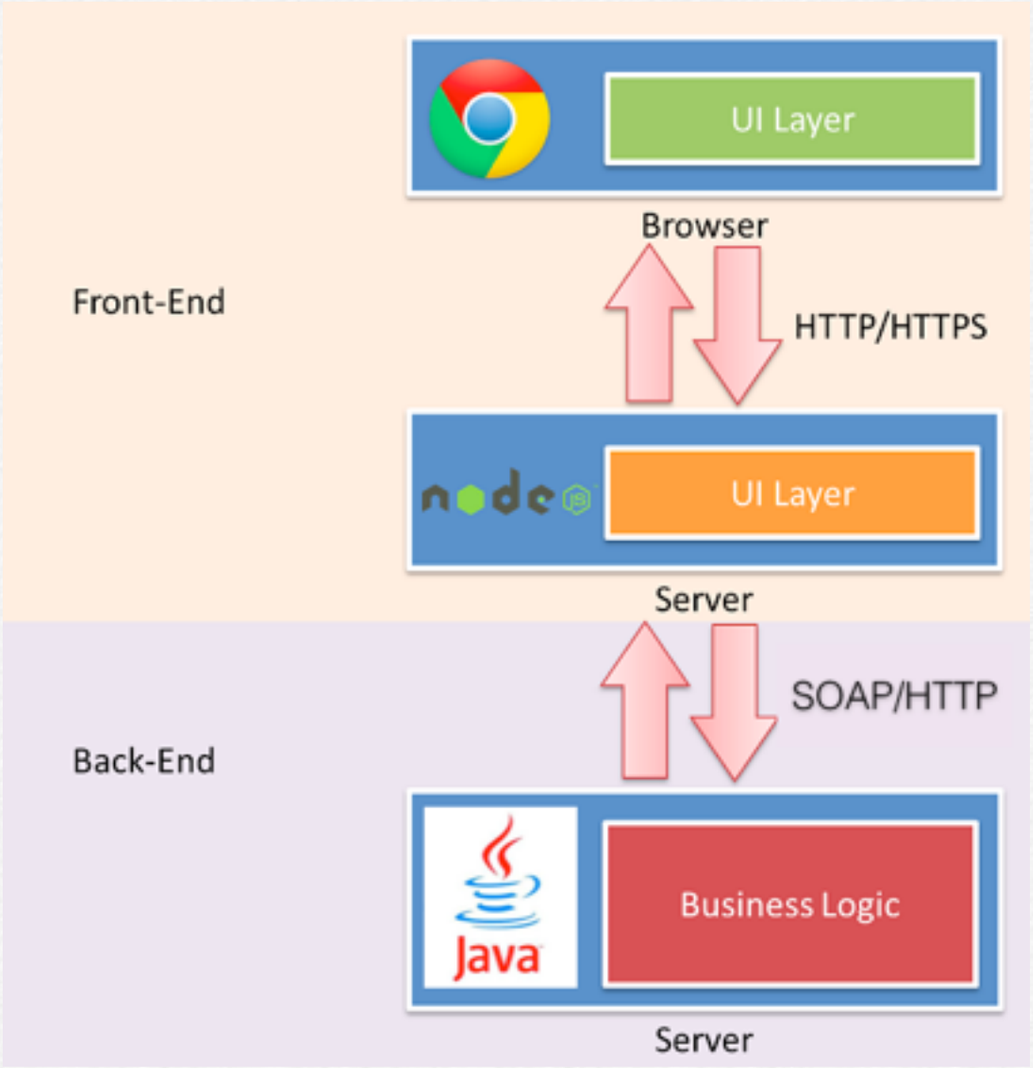


试想一下，如果前端掌握了Controller，我们可以做url design，我们可以根据场景决定在服务端同步渲染，还是根据view层数据

输出json数据，我们还可以根据表现层需求很容易的做 big-pipe,comet,socket等等，完全是需求决定使用方式。

3.1 基于NodeJS“全栈”式开发

如果想实现上图的分层，就必然需要一种web服务帮我们实现以前后端做的事情，于是就有了标题提到的“基于NodeJS的全栈式开发”



这张图看起来简单而且很好理解，但没尝试过，会有很多疑问。

- SPA模式中，后端已供了所需的数据接口，view前端已经可以控制，为什么要多加NodeJS这一层?
- 多加一层，性能怎么样?
- 多加一层，前端的工作量是不是增加了?
- 多加一层就多一层风险，怎么破?
- NodeJS什么都能做，为什么还要JAVA?

这些问题要说清楚不容易，下面说下我的认识过程。

3.2 为什么要增加一层NodeJS?

现阶段我们主要以后端MVC的模式进行开发，这种模式严重阻碍了前端开发效率，也让后端不能专注于业务开发。
解决方案是让前端能控制Controller层，但是如果在现有技术体系下很难做到，因为不可能让所有前端都学java，安装后端的开发环境，写VM。
NodeJS就能很好的解决这个问题，我们无需学习一门新的语言，就能做到以前开发帮我们做的事情，一切都显得那么自然。

3.3 性能问题

分层就涉及每层之间的通讯，肯定会有一定的性能损耗。但是合理的分层能让职责清晰、也方便协作，会大大提高开发效率。分层带来的损失，一定能在其他方面的收益弥补回来。

另外，一旦决定分层，我们可以通过优化通讯方式、通讯协议，尽可能把损耗降到最低。

举个例子：
detail静态化之后，还是有不少需要实时获取的信息，比如优化、物流、促销等等，因为这些信息在不同业务系统中，所以需要前端发送5，6个异步请求来回填这些内容。
有了NodeJS之后，前端可以在NodeJS中去代理这5个异步请求，还能很容易的做bigpipe,这块的优化能让整个渲染效率提升很多。
可能在PC上你觉得发5,6个异步请求也没什么，但是在无线端，在客户手机上建立一个http请求开销很大，有了这个优化，性能一下提升好几倍。

淘宝详情基于NodeJS的优化我们正在进行中，上线之后我会分享一下优化的过程。

3.4 前端的工作量是否增加了?

相对于只切页面/
做demo，肯定是增加了一点，但是当前模式下有联调、沟通环节，这个过程非常花时间，也容易出bug，还很难维护。
所以，虽然工作量会增加一点，但是总体开发效率会提升很多。

另外，测试成本可以节省很多。以前开发的接口都是针对表现层的，很难写测试用例。如果做了前后端分离，甚至测试都可以分开，一拨人专门测试接口，一拨人专注测试UI（这部分工作甚至可以用工具代替）。

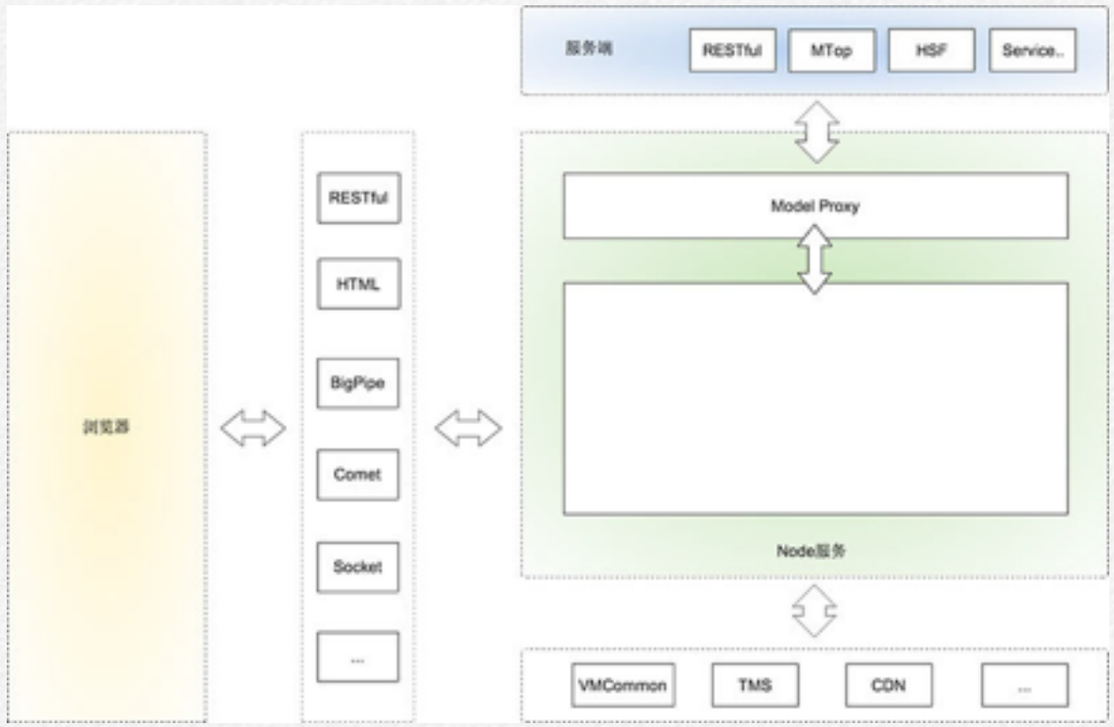
3.5 增加Node层带来的风险怎么控制?

这个担心有点多余，随着Node大规模使用，SCM/PE/
安全，都会解决进行。
他们会帮助我们在申请，测试，发布，风控几个层面保证稳定性

3.6 Node什么都能做，为什么还要JAVA?

我们的初衷是做前后端分离，如果考虑这个问题就有点违背我们的初衷了。即使用Node替代Java，我们也没办法保证不出现今天遇到的种种问题，比 如职责不清。我们的目的是分层开发，专业的人，专注做专业的事。基于JAVA的基础架构已经非常强大而且稳定，而且更适合做现在架构的事情。

四、淘宝基于Node的前后端分离



上图是我理解的淘宝基于Node的前后端分离分层，以及Node的职责范围。简单解释下：

- 最上端是服务端，就是我们常说的后端。后端对于我们来说，就是一个接口的集合，服务端提供各种各样的接口供我们使用。因为有Node层，也不用局限是什么形式的服务。对于后端开发来说，他们只用关心业务代码的接口实现。

- 服务端下面是Node应用。

- Node应用中有一层Model Proxy与服务端进行通讯。这一层主要目前是抹平我们对不同接口的调用方式，封装一些view层需要的Model。

- （Node层还能轻松实现原来vmcommon,tms等需要）

- Node层要使用什么框架由开发者自己决定。不过推荐使用express+xTemplate的组合，xTemplate能做到前后端公用。

- 怎么用Node大家自己决定，但是令人兴奋的是，我们终于可以使用Node轻松实现我们想要的输出方式:JSON/JSONP/RESTful/HTML/BigPipe/Comet/Socket/同步、异步，想怎么整就怎么整，完全根据你的场景决定。

- 浏览器层在我们这个架构中没有变化，也不希望因为引入Node改变你以前在浏览器中开发的认知。

- 引入Node，只是把本该就前端控制的部分交由前端掌控。

这种模式我们已经有两个项目在开发中，虽然还没上线，但是无论是在开发效率，还是在性能优化方面，我们都已经尝到了甜头。

五、我们还需要要做什么?

- 把Node的开发流程集成到淘宝现有的SCM流程中。

- 基础设施建设，比如session,logger等通用模块。

- 最佳开发实践

- 线上成功案例

- 大家对Node前后端分离概念的认识

- 安全

- 性能

- …

技术上不会有太多需要去创新和研究的，已经有非常多现成的积累。其实关键是一些流程的打通和通用解决方案的积累，相信随着更多的项目实践，这块慢慢会变成一个稳定的流程。

六、“中途岛”

虽然“基于NodeJS的全栈式开发”模式很让人兴奋，但是把基于Node的全栈开发变成一个稳定，让大家都能接受的东西还有很

多路要走，我们正在进行的“中途岛”项目就是为了解决这个问题。虽然我们起步不久，但是离目标已经越来越近！！

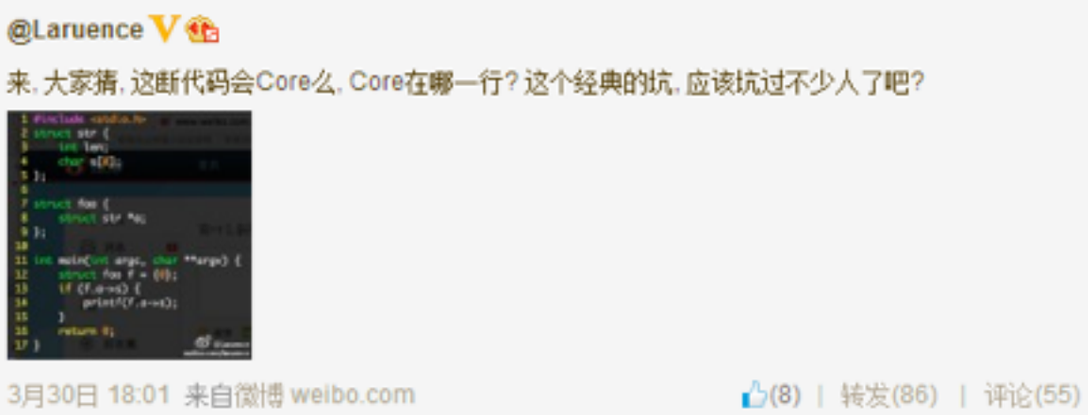
作者：常胤

原文链接：

http://ued.taobao.org/blog/2014/04/%E4%B9%9F%E8%B0%88%E5%9F%BA%E4%BA%8Enodejs%E7%9A%84%E5%85%A8%E6%A0%88%E5%BC%8F%E5%BC%80%E5%8F%91%EF%BC%88%E5%9F%BA%E4%BA%8Enodejs%E7%9A%84%E5%89%8D%E7%AB%AF%E5%90%8E%E7%AB%AF%E5%88%86%E7%A6%BB/

C语言结构体里的成员数组和指针

单看这文章的标题，你可能会觉得好像没什么意思。你先别下这个结论，相信这篇文章会对你理解C语言有帮助。这篇文章产生的背景是在微博上，看到@Laruence同学出了一个关于C语言的题，微博链接。微博截图如下。我觉得好多人对这段代码的理解还不够深入，所以写下了这篇文章。



为了方便你把代码copy过去编译和调试，我把代码列在下面：

```
#include <stdio.h>
struct str{
    int len;
    char s[0];
};

struct foo {
    struct str *a;
};

int main(int argc, char** argv) {
    struct foo f={0};
    if (f.a->s) {
        printf( f.a->s);
    }
    return 0;
}
```

你编译一下上面的代码，在VC++和GCC下都会在14行的printf处crash掉你的程序。@Laruence 说 这个是个经典的坑，我觉得这怎么会是经典的坑呢？上面这代码，你一定会问，为什么if语句判断的不是f.a？而是f.a里面的数组？写这样代码的人脑子里 在想什么？还是用这样的代码来玩票？不管怎么样，看过原微博的回复，我个人觉得大家主要还是对C语言理解不深，如果这算坑的话，那么全都是坑。

接下来，你调试一下，或是你把14行的printf语句改成：

```
printf("%x\n", f.a->s);
```

你会看到程序不crash了。程序输出：4。 这下你知道了，访问0×4的内存地址，不crash才怪。于是，你一定会有如下的问题：

1）为什么不是 13行if语句出错？f.a被初始化为空了嘛，用空指针访问成员变量为什么不crash？

2）为什么会访问到了0×4的地址？靠，4是怎么出来的？

3）代码中的第4行，char s[0] 是个什么东西？零长度的数组？为什么要这样玩？

让我们从基础开始一点一点地来解释C语言中这些诡异的问题。

结构体中的成员

首先，我们需要知道——**所谓变量，其实是内存地址的一个抽像名字罢了**。在静态编译的程序中，所有的变量名都会在编译时被转成内存地址。机器是不知道我们取的名字的，只知道地址。

所以有了——栈内存区，堆内存区，静态内存区，常量内存区，我们代码中的所有变量都会被编译器预先放到这些内存区中。

有了上面这个基础，我们来看一下结构体中的成员的地址是什么？我们先简单化一下代码：

```
1  struct test{
2      int i;
3      char *p;
4  };
```

上面代码中，test结构中i和p指针，在C的编译器中保存的是相对地址——也就是说，他们的地址是相对于struct test的实例的。如果我们有这样的代码：

```
1  struct test t;
```

我们用gdb跟进去，对于实例t，我们可以看到：

```
# t实例中的p就是一个野指针
(gdb) p t
$1 = {i = 0, c = 0 '\000', d = 0 '\000', p = 0x4003e0 "1\355I\211\..."}

# 输出t的地址
(gdb) p &t
$2 = (struct test *) 0x7fffffffe5f0

#输出(t.i)的地址
(gdb) p &(t.i)
$3 = (char **) 0x7fffffffe5f0

#输出(t.p)的地址
(gdb) p &(t.p)
$4 = (char **) 0x7fffffffe5f4
```

我们可以看到，t.i的地址和t的地址是一样的，t.p的址址相对于t的地址多了个4。说白了，**t.i 其实就是(&t + 0×0), t.p 的其实就是 (&t + 0×4)**。0×0和0×4这个偏移地址就是成员i和p在编译时就

被编译器给hard code了的地址。于是，你就知道，**不管结构体的实例是什么——访问其成员其实就是加成员的偏移量。**

下面我们来做个实验：

```
struct test{
    int i;
    short c;
    char *p;
};

int main(){
    struct test *pt=NULL;
    return 0;
}
```

编译后，我们用gdb调试一下，当初始化pt后，我们看看如下的调试：（我们可以看到就算是pt为NULL，访问其中的成员时，其实就是在访问相对于pt的内址）

```
(gdb) p pt
$1 = (struct test *) 0x0
(gdb) p pt->i
Cannot access memory at address 0x0
(gdb) p pt->c
Cannot access memory at address 0x4
(gdb) p pt->p
Cannot access memory at address 0x8
```

注意：上面的pt->p的偏移之所以是0×8而不是0×6，是因为内存对齐了（我在64位系统上）。关于内存对齐，可参看《深入理解C语言》一文。

好了，现在你知道为什么原题中会访问到了0×4的地址了吧，因为是相对地址。

相对地址有很好多处，其可以玩出一些有意思的编程技巧，比如把C搞出面向对象式的感觉来，你可以参看我正好11年前的文章《用C写面向对像的程序》（用指针类型强转的危险玩法——相对于C++来说，C++编译器帮你管了继承和虚函数表，语义也清楚了很多）

指针和数组的差别

有了上面的基础后，你把源代码中的struct str结构体中的char s[0];改成char *s;试试看，你会发现，在13行if条件的时候，程序因为Cannot access memory就直接挂掉了。为什么声明成char s[0]，程序会在14行挂掉，而声明成char *s，程序会在13行挂掉呢？**那么char *s 和 char s[0]有什么差别呢?**

在说明这个事之前，有必要看一下汇编代码，用GDB查看后发现：

- 对于char s[0]来说，汇编代码用了lea指令，lea 0×04(%rax)， %rdx
- 对于char*s来说，汇编代码用了mov指令，mov 0×04(%rax)， %rdx

lea全称load effective address，是把地址放进去，而mov则是把地址里的内容放进去。所以，就crash了。

从这里，我们可以看到，访问成员数组名其实得到的是数组的相对地址，而访问成员指针其实是相对地址里的内容（这和访问其它非指针或数组的变量是一样的）

换句话说，**对于数组 char s[10]来说，数组名 s 和 &s 都是一样的**（不信你可以自己写个程序试试）。在我们这个例子中，也就是说，都表示了偏移后的地址。这样，如果我们访问 指针的地址（或是成员变量的地址），那么也就不会让程序挂掉了。

正如下面的代码，可以运行一点也不会crash掉（你汇编一下你会看到用的都是lea指令）：

```
struct test{
    int i;
    short c;
    char *p;
    char s[10];
};

int main(){
    struct test *pt=NULL;
    printf("&s = %x\n", pt->s); //等价于 printf("%x\n", &(pt->s) );
    printf("&i = %x\n", &pt->i); //因为操作符优先级，我没有写成&(pt->i)
    printf("&c = %x\n", &pt->c);
    printf("&p = %x\n", &pt->p);
    return 0;
}
```

看到这里，你觉得这能算坑吗？不要出什么事都去怪语言，大家要想想是不是问题出在自己身上。

关于零长度的数组

首先，我们要知道，0长度的数组在ISO C和C++的规格说明书中是不允许的。这也就是为什么在VC++2012下编译你会得到一个警告：“arning C4200: 使用了非标准扩展：结构/联合中的零大小数组”。

那么为什么gcc可以通过而连一个警告都没有？那是因为gcc 为了预先支持C99的这种玩法，所以，让“零长度数组”这种玩法合法了。关于GCC对于这个事的文档在这里：“Arrays of Length Zero”，文档中给了一个例子（我改了一下，改成可以运行的了）：

```
struct test{
    int i;
    short c;
    char *p;
    char s[10];
};

int main(){
    struct test *pt=NULL;
    printf("&s = %x\n", pt->s); //等价于 printf("%x\n", &(pt->s) );
    printf("&i = %x\n", &pt->i); //因为操作符优先级，我没有写成&(pt->i)
    printf("&c = %x\n", &pt->c);
    printf("&p = %x\n", &pt->p);
    return 0;
}
```

上面这段代码的意思是：我想分配一个不定长的数组，于是我有一个结构体，其中有两个成员，一个是length，代表数组的长度，一个是 contents，代码数组的内容。后面代码里的this_length（长度是10）代表是我想分配的数据的长度。（这看上去是不是像一个C++的类？）这种玩法英文叫：Flexible Array，中文翻译叫：柔性数组。

我们来用gdb看一下：

```
(gdb) p thisline
$1 = (struct line *) 0x601010

(gdb) p *thisline
$2 = {length = 10, contents = 0x601010 "\n"}

(gdb) p thisline->contents
$3 = 0x601014 "aaaaaaaaaa"
```

我们可以看到：在输出*thisline时，我们发现其中的成员变量contents的地址居然和thisline是一样的（偏移量为 0×0??!!）。但是当我们输出thisline->contents的时候，你又发现contents的地址是被offset了0×4了的，内 容也变成了10个‘a’。（我觉得这是一个GDB的bug，VC++的调试器就能很好的显示）

我们继续，如果你sizeof(char[0])或是 sizeof(int[0]) 之类的零长度数组，你会发现sizeof返回了0，这就是说，零长度的数组是存在于结构体内的，但是不占结构体的size。你可以简单的理解为一个没有内 容的占位标识，直到我们给结构体分配了内存，这个占位标识才变成了一个有长度的数组。

看到这里，你会说，为什么要这样搞啊，把contents声明成一个指针，然后为它再分配一下内存不行么？就像下面一样。

```
struct line {
   int length;
   char *contents;
};

int main(){
    int this_length=10;
    struct line *thisline = (struct line *)malloc (sizeof (struct line));
    thisline->contents = (char*) malloc( sizeof(char) * this_length );
    thisline->length = this_length;
    memset(thisline->contents, 'a', this_length);
    return 0;
}
```

这不一样清楚吗？而且也没什么怪异难懂的东西。是的，这也是普遍的编程方式，代码是很清晰，也让人很容易理解。即然这样，那为什么要搞一个零长度的数组？有毛意义？！

这个事情出来的原因是——**我们想给一个结构体内的数据分配一个连续的内存**！这样做的意义有两个好处：

第一个意义是，方便内存释放。如果我们的代码是在一 个给别人用的函数中，你在里面做了二次内存分配，并把整个结构体返回给用户。用户调用free可以释放结构体，但是用户并不知道这个结构体内的成员也需要 free，所以你不能指望用户来发现这个事。所以，如果我们把结构体的内存以及其成员要的内存一次性分配好了，并返回给用户一个结构体指针，用户做一次 free就可以把所有的内存也给释放掉。（读到这里，你一定会觉得C++的封闭中的析构函数会让这事容易和干净很多）

第二个原因是，这样有利于访问速度。连续的内存有益于提高访问速度，也有益于减少内存碎片。（其实，我个人觉得也没多高了，反正你跑不了要用做偏移量的加法来寻址）

我们来看看是怎么个连续的，用gdb的x命令来查看：(我们知道，用struct line {}中的那个char contents[]不占用结构体的内存，所以，struct line就只有一个int成员，4个字节，而我们还要为contents[]分配10个字节长度，所以，一共是14个字节)

```
(gdb) x /14b thisline
0x601010:       10      0       0       0       97      97      97      97
0x601018:       97      97      97      97      97      97
```

从上面的内存布局我们可以看到，前4个字节是 int length，后10个字节就是char contents[]。

如果用指针的话，会变成这个样子：

```
(gdb) x /16b thisline
0x601010:       1       0       0       0       0       0       0       0
0x601018:       32      16      96      0       0       0       0       0
(gdb) x /10b this->contents
0x601020:       97      97      97      97      97      97      97      97
0x601028:       97      97
```

上面一共输出了四行内存，其中，

- 第一行前四个字节是 int length，第一行的后四个字节是对齐。
- 第二行是char* contents，64位系统指针8个长度，他的值是0×20 0×10 0×60 也就是0×601020。
- 第三行和第四行是char* contents指向的内容。

从这里，我们看到，**其中的差别——数组的原地就是内容，而指针的那里保存的是内容的地址。**

后记

好了，我的文章到这里就结束了。但是，请允许我再唠叨两句。

1）看过这篇文章，你觉得C复杂吗？我觉得并不简单。某些地方的复杂程度不亚于C++。

2）那些学不好C++的人一定是连C都学不好的人。连C都没学好，你们根本没有资格鄙视C++。

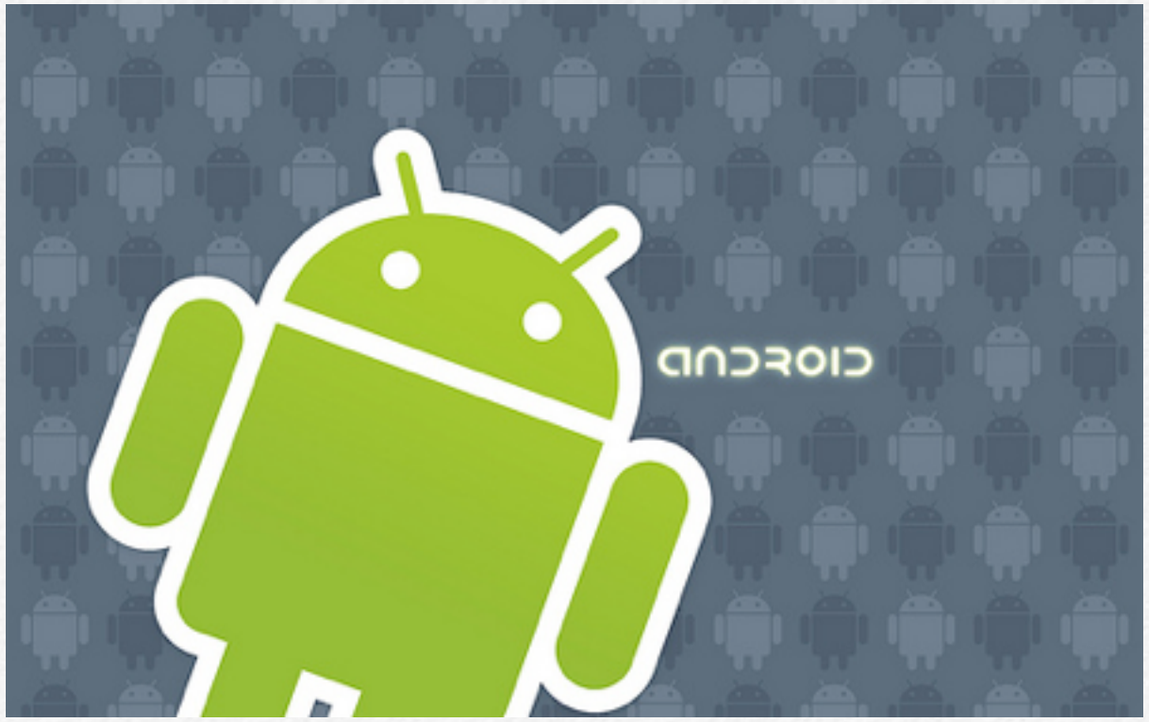
3）当你们在说有坑的时候，你得问一下自己，是真有坑还是自己的学习能力上出了问题。

如果你觉得你的C语言还不错，欢迎你看看《C语言的谜题》还有《谁说C语言很简单？》还有《语言的歧义》以及《深入理解C语言》一文。

作者：陈皓

原文出处：**酷 壳 – CoolShell.cn**

高效开发Android App的10个建议



假如要Google Play上做一个最失败的案例，那最好的秘诀就是界面奇慢无比、耗电、耗内存。接下来就会得到用户的消极评论，最后名声也就臭了。即使你的应用设计精良、创意无限也没用。

耗电或者内存占用等影响产品效率的每一个问题都会影响App的成功。这就是为什么在开发中确保最优化、运行流畅而且不会使Android系统出问题 是至关重要的了。这里不需要讨论高效编程，因为我们不会关心你写的代码是否能够经得起测试。即使高效的代码也是需要时间来运行。今天这篇文章我们就讲讲怎 么尽可能地缩短运行时间，以及如何开发用户喜欢的App。

高效地利用线程

建议一：怎么在后台取消一些线程中的动作

我们知道App运行过程中所有的操作都默认在主线程（UI线程）中进行的，这样App的响应速度就会受到影响。会导致程序陷入卡顿、死掉甚至会发生系统错误。

为了加快响应速度，需要把费时的操作（比如网络请求、数据库操作或者复杂的计算）从主线程移动到一个单独的线程中。最高效的方式就是在类这一级完成 这项操作，可以使用AsyncTask或者IntentService来创建后台操作。如果选择使用IntentService，它会在需要的时候启动起 来，然后通过一个工作线程来处理请求（Intent）。

使用IntentService时需要注意以下几点限制：

- 这个类不要给UI传递信息，如果要向用户展示处理结果信息请用Activity；
- 每次只能处理一个请求；
- 每一个处理请求过程都不能中断；

建议二：怎么保持响应不发生ANR

从UI线程中移除费时操作这个方式还可以防止用户操作出现系统不响应（ANR）对话框。需要做的就是继承AsyncTask来创建一个后台工作线程，并实现doInBackground()方法。

还有一种方式就是自己创建一个Thread类或者HandlerThread类。需要注意这样也会使App变慢，因为默认的线程优先级和主线程的优先级是一样的，除非你明确设定线程的优先级。

建议三：怎么在线程中初始化查询操作

当查询操作正在后台处理时，展示数据也不是即时的，但是你可以使用CursorLoader对象来加快速度，这个操作可以使Activity和用户之间的互动不受影响。

使用这个对象后，你的App会为ContentProvider初始化一个独立的后台线程进行查询，当查询结束后就会给调用查询的Activity返回结果。

建议四：其它需要注意的方面

- 使用StrictMode来检查UI线程中可能潜在的费时操作；
- 使用一些特殊的工具如Systrace或者Trace-view来寻找在你的应用中的瓶颈；
- 用进度条向用户展示操作进度；
- 如果初始化操作很费时，请展示一个欢迎界面。

优化设备的电池寿命

如果应用很费电，请不要责怪用户卸载了你的应用。对于电池使用来说，主要费电情况如下：

- 更新数据时经常唤醒程序；
- 用EDGE或者3G来传递数据；
- 文本数据转换，进行非JIT正则表达式操作。

建议五：怎么优化网络

- 如果没有网络连接，请让你的应用跳过网络操作；只在有网络连接并且无漫游的情况下更新数据；
- 选择兼容的数据格式，把含有文本数据和二进制数据的请求全部转化成二进制数据格式请求；
- 使用高效的转换工具，多考虑使用流式转换工具，少用树形的转换工具；
- 为了更快的用户体验，请减少重复访问服务器的操作；
- 如果可以的话，请使用framework的G-ZIP库来压缩文本数据以高效使用CPU资源。

建议六：怎么优化应用在前端的工作

- 如果考虑使用wakelocks，尽量设置为最小的级别；
- 为了防止潜在的bug导致的电量消耗，请明确指定超时时间；
- 启用 android:keepScreenOn属性；
- 除了系统的GC操作，多考虑手动回收Java对象，比如XmlPullParserFactory和BitmapFactory。还有正则表达式的Matcher.reset(newString)操作、StringBuilder.setLength(0)操作；
- 要注意同步的问题，尽管在主线程中是安全的；
- 在Listview中要多采用重复利用策略；
- 如果允许的话多使用粗略的网络定位而不用GPS，对比一下GPS需要1mAh（25s * 140 mA），而一般网络只用0.1mAh（2s * 180mA）；
- 确保注销GPS的位置更新操作，因为这个更新操作在onPause()中也是会继续的。当所有的应用都注销了这个操作，用户可以在系统设置中重新启用GPS而不浪费电量；
- 请考虑在大量数理运算中使用低精度变量并在用DisplayMetrics进行DPI任务时缓存变量值；

建议七：怎么优化工作在前台的应用

- 请确保service生命周期都是短暂的，因为每个进程都需要2MB的内存，而在前台程序需要内存时也会重新启动；
- 保持内存的使用量不要太大；
- 如果要应用每30分钟更新一次，请在设备处于唤醒状态下进行；

- Service在pull或者sleep状态都是不好的，这就是为什么在服务结束时要使用AlarmManager或者配置属性stopSelf()的原因。

建议八：其它注意事项

- 在进行整体更新之前检查电池的状态和网络状态，等待最好的状态在进行大幅度装换操作；
- 让用户看到用电情况，比如更新周期，后台操作的时候；

实现低内存占用UI

建议九：怎么找到布局显示问题

当我们为布局单独创建UI的时候，就是在创建滥用内存的App，它在UI中会出现可恶的延时。要实现一个流畅的、低内存占用的UI，第一步就是搜索 你的应用找出潜在的瓶颈布局。使用Android SDK/tools/中自带的Hierarchy Viewer Tool工具。

还有一个很好的工具就是Lint，它会扫描应用的源码去寻找可能存在的bug，并为控件结果进行优化。

建议十：如何解决问题

如果布局显示结果发现了问题，你可以考虑简化布局结构。可以把LinearLayout类型转化成RelativeLayout类型，降低布局的层级结构。

做到更加完美并不断优化

尽管以上的每条建议看起来都是很小的改进，但是如果它能成为你日常代码的一部分，那么你就会看到意想不到的结果。要让Google Play看到更多杰出的、流畅的、更快速、更省电的应用，向Android走向完美的目标迈进一步。

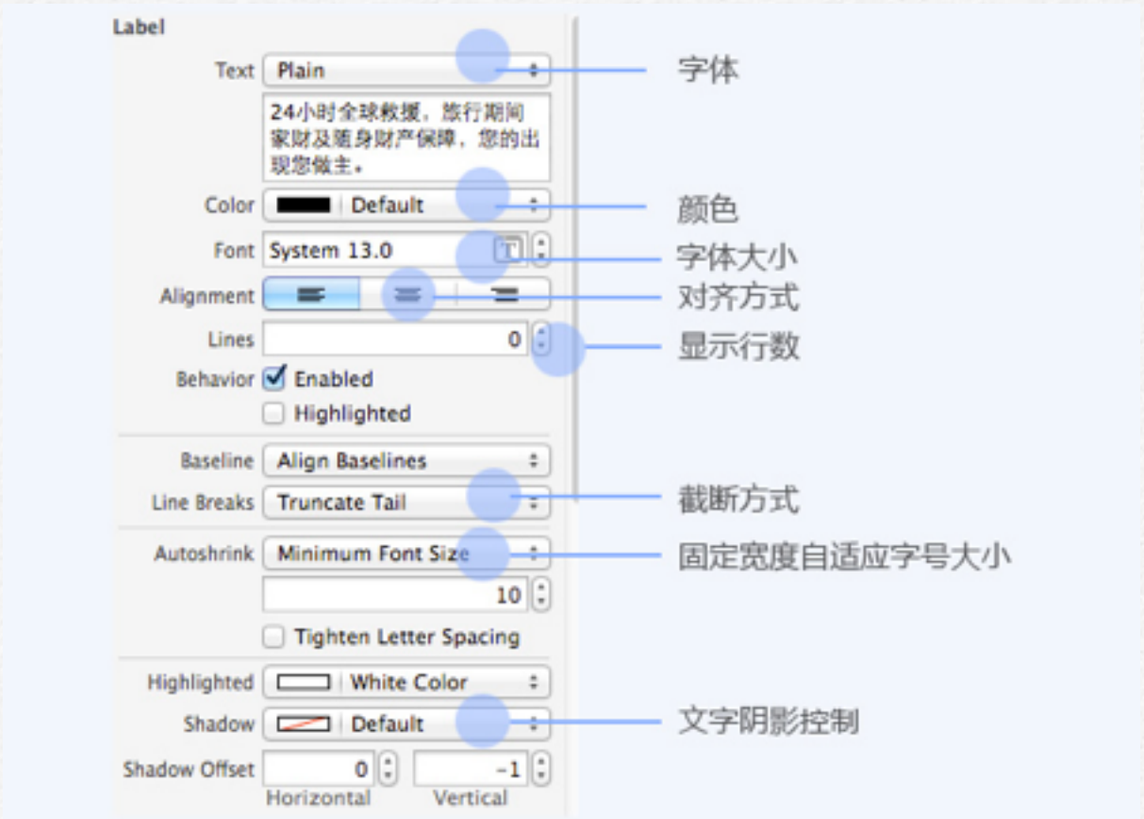
原文链接： azoft 翻译： 伯乐在线 - chris
译文链接： http://blog.jobbole.com/64279/

移动APP的文本样式控制

我们能够看到精美的网页，是因为浏览器支持CSS样式；但是在APP开发工具里面，文本样式的控制还停留在字号大小层面上，CSS可以简单控制的文 本样式想要在APP里面显示的一样漂亮并非易事。而如今手机平板已成为我们主要的阅读设备，只要涉及资讯阅读的APP都需要解决这一问题，本篇就简单介绍 下主流APP是如何来控制文本样式的：

- **简单粗暴的使用开发工具控制**

文本展示最简单粗暴的就是用程序语言直接呈现文本，在OS X平台下的开发工具Xcode主要的文本样式控制包括：



如上图所示，原生iOS应用的文本样式控制缺少常用的文字行间距控制，而且只能以整体区域为单位控制样式。而Android开发工具下使用 TextView理论上可以写出更多的文本样式，但也比较难控制。使用程序语言来实现文本样式只能适和较少文字的展示，当需要展示较多文字时，则会相当难 看，比如豆瓣电影：

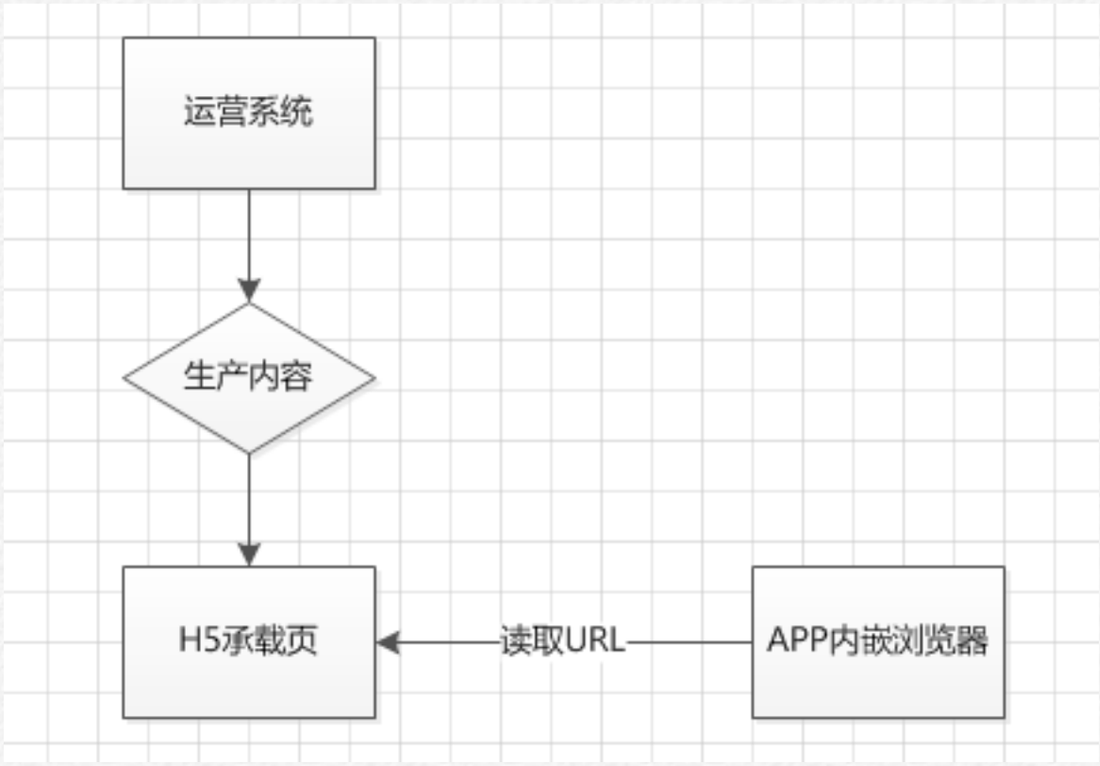
豆瓣电影作为一款知名应用，使用这样简单粗暴的文本实现方式实在愧对这么多的忠实用户。

- **H5承载页 + WebView**

鉴于原生开发工具在文本样式上控制的局限性，行业内成熟的APP应用都采用了Native+Web的混合视图，即在APP中嵌入浏

览器框架，加载HTML网页。这样就可以完美的呈现富文本样式。比如微信的自媒体：

这样的方式需要建立独立的H5站点，考虑到资讯页面一般都需要以链接的方式分享到站外，所以大都数阅读类的APP都是同时开发H5站和APP，利用一个后台提供内容给H5站和APP;业务模型如下图所示：

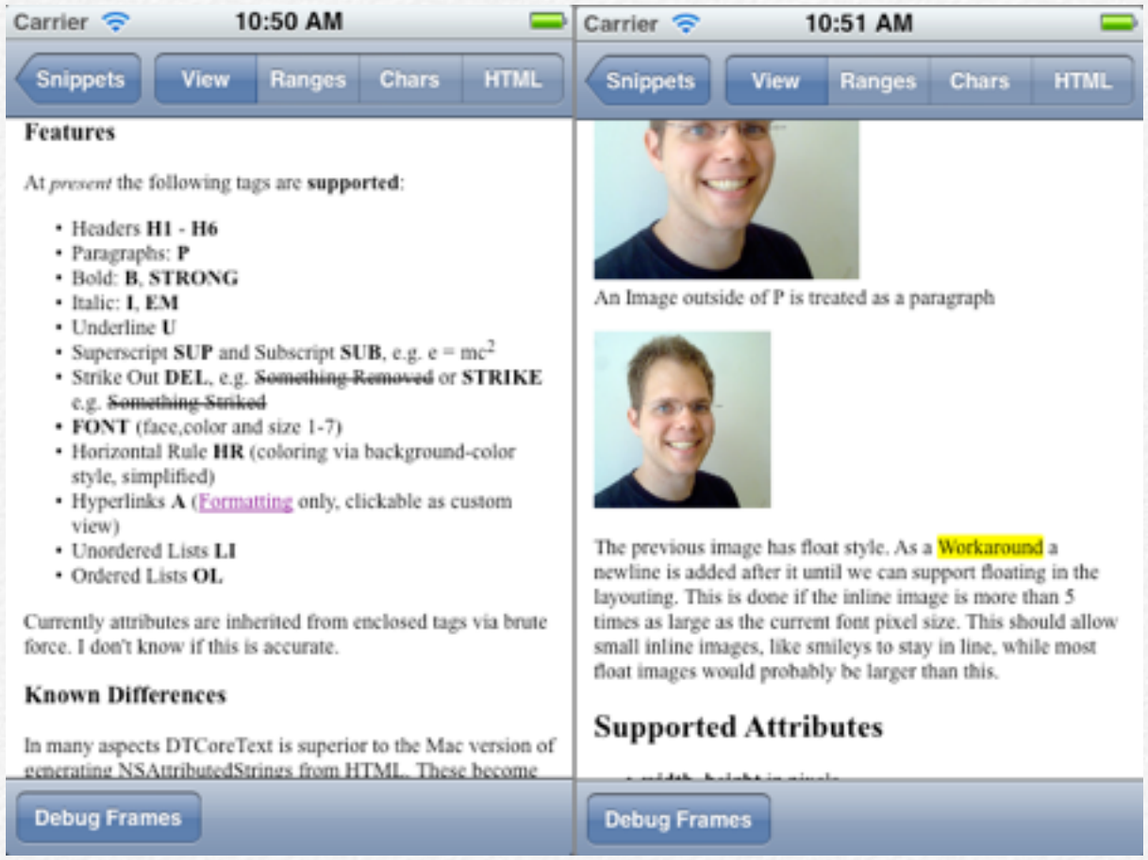


在APP中嵌入浏览器加载内容唯一的缺点就是加载速度相对慢一些。但普通用户基本上看不出区别，所以是当前最流行的实现方式；

- **使用DTCoreText等文字效果代码类库（iOS平台）**

H5承载页 + 内嵌浏览器框架的方式虽然可以完美的呈现文本内容，但加载速度会慢一些，而且展示风格会有一点点突兀；那么还有一种方法就是使用DTCoreText;

DTCoreText 是一个功能十分强大的文字效果代码类库。在UITextView上实现十分丰富的文字效果，包括文字大小、颜色、字体、下划线，链接，给文字加上图片、视频，文字任意间距等等。实现类似于CSS网页的文字效果。



使用这种实现方法的APP比如知乎，同样是以文字评论为主的APP，用户体验相比豆瓣电影就立刻显得高大上了：

这种方式可以直接实现富文本效果，但鉴于应用本身也需要以外链的方式分享到SNS站点，所以同样需要建立独立的H5站点；

本篇的主要目的是想让产品角色或设计角色对这方面的知识有个初步了解。因为本人非开发出身，写这篇文章时基本上也是参考网上的一零散资料，如有错误，欢迎指出...

作者：xiaomeng小猛

原文链接：http://www.54xiaomeng.com/?p=943

Ceph —— 一个 PB 规模的 Linux 分布式文件系统

Linux 持续进军可伸缩计算领域，尤其是可扩展存储领域。 Linux 文件系统最近新增了一个引人注目的选择 —— Ceph，一个维持 POSIX 兼容的同时还集成了复制、容错的分布式文件系统。探讨 Ceph 的架构，可以知道它是如何提供容错性，及如何简化大量数据的管理的。

作为存储行业的架构师，我对文件系统情有独钟。这些系统是存储系统的用户接口，尽管它们都倾向于提供相似的特性集，它们还是会提供引人注目的不同点。Ceph 也如此，它提供了一些你会在文件系统中找到的最有趣的特性。

Ceph 始于加州大学圣克鲁兹分校的 Sage Weil 的博士学位课题。但从 2010 年 3 月下旬起，你可以 Linux 主流内核中找到 Ceph （从 2.6.34 内核起）。

尽管 Ceph 可能还未对生产环境做好准备，但它依然有助于评估目的。本文探讨了 Ceph 文件系统，及其成为可扩展分布式存储的诱人选择的独到之处。

Ceph 目标

开发文件系统是一种复杂的投入，但是如果能准确的解决问题的话，则拥有着不可估量的价值。Ceph 的目标可以简单的定义为：

- 容易扩展到 PB 量级
- 不同负荷下的高性能 （每秒输入输出操作数 [IPOS]、带宽）
- 可靠性高

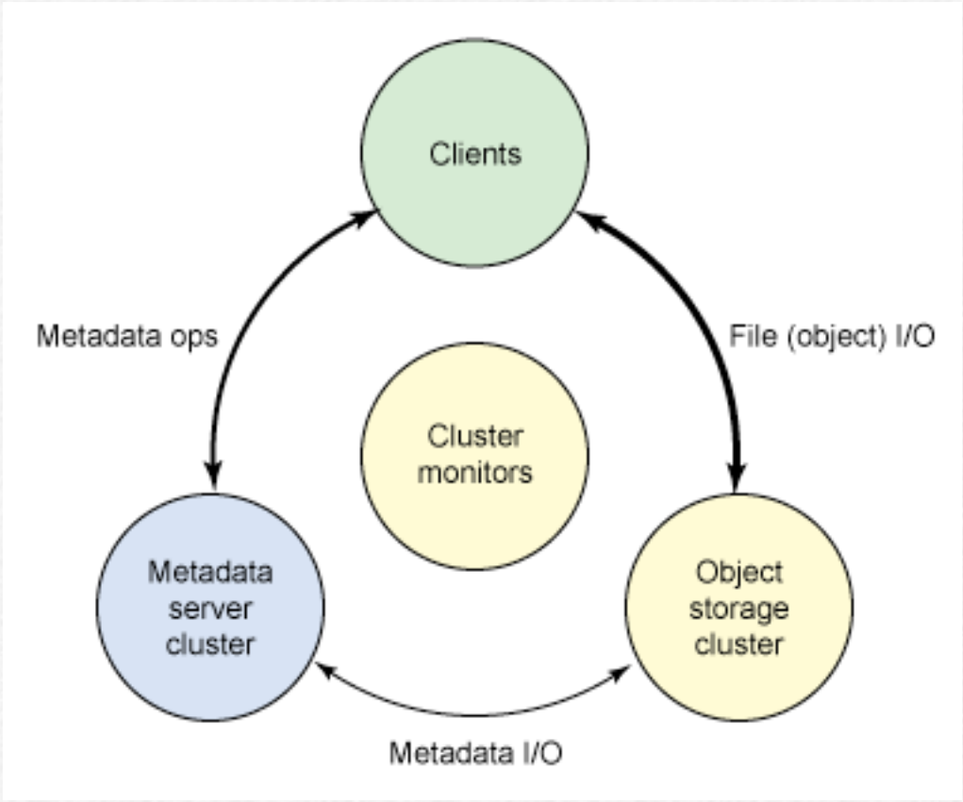
不幸的，这些目标彼此间矛盾（例如，可扩展性会减少或阻碍性能，或影响可靠性）。Ceph 开发了一些有趣的概念（例如动态元数据分区、数据分布、复制），本文会简单探讨。Ceph 的设计也集成了容错特性来防止单点故障，并假定，大规模（PB 级的存贮）的存储故障是一种常态，而非异常。最后，它的设计没有假定特定的工作负荷，而是包含了可变的分布式工作负荷的适应能力，从而提供最佳的性能。它 以 POSIX 兼容为目标完成这些工作，允许它透明的部署于那些依赖于 POSIX 语义的现存应用（通过 Ceph 增强功能）。最后，Ceph 是开源分布式存储和 Linux 主流内核的一部分（2.6.34）。

Ceph 架构

现在，让我们先在上层探讨 Ceph 架构及其核心元素。之后深入到其它层次，来辨析 Ceph 的一些主要方面，从而进行更详细的分析。

Ceph 生态系统可以大致划分为四部分（见图1）：客户端（数据使用者）、元数据服务器（缓冲及同步分布的元数据）、对象存储集群（以对象方式存储数据与元数据，实现其它主要职责），及集群监控（实现监控功能）。

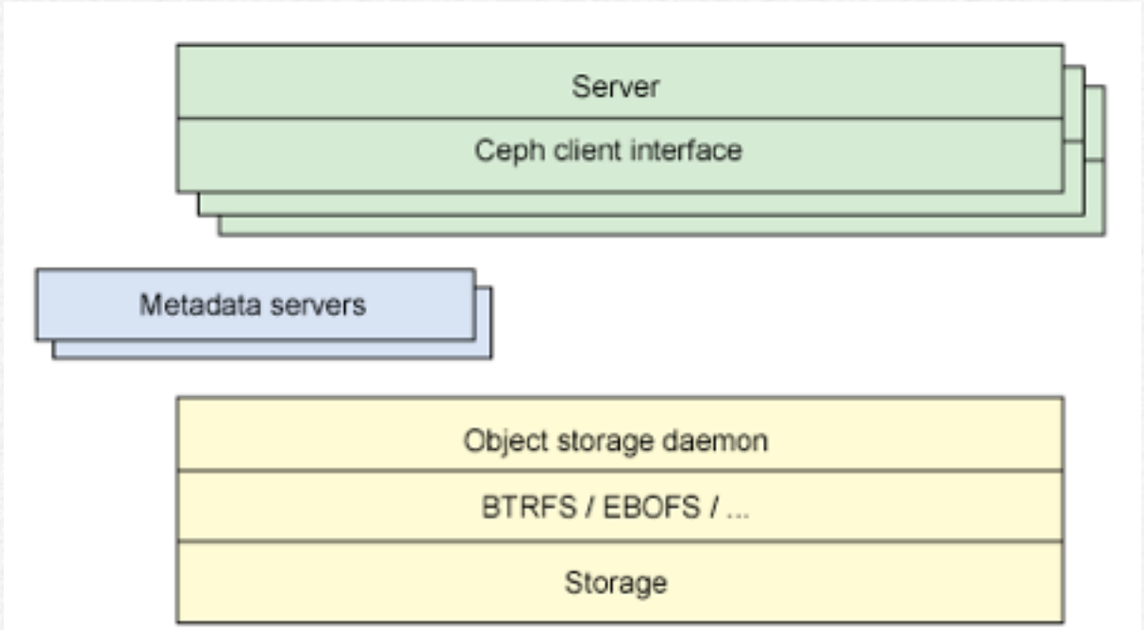
图 1. Ceph 生态系统的概念架构



如图一所示，客户端通过元数据服务器来执行元数据操作（以识别数据位置）。元数据服务器管理数据的位置及新数据的存储位置。注意，元数据存储在存储 集群中（如 "Metadata I/O" 所示）。真正的文件 I/O 发生在客户端和对象存储集群之间。以这种方式，较高级的 POSIX 功能（如 open、close、re-name）由元数据服务器管理，与此同时，POSIX 功能（如 read、write）直接由对象存储集群管理。

图2 提供了架构的另一视角图。一组服务器通过客户端接口访问 Ceph 生态系统，接口理解元数据服务器和对象级存储的关系。这种分布式存储系统可被看作为几层，包括存储设备的格式（扩展的和基于 B-树 的对象文件系统[ebofs]，或者其它备选），为管理数据复制、故障检测、恢复、后续数据迁移（称之为可靠的自发分布式对象存储，RADOS）等所设计 的至关重要的管理层。最后，监控用于鉴定组件故障，包括后续通知。

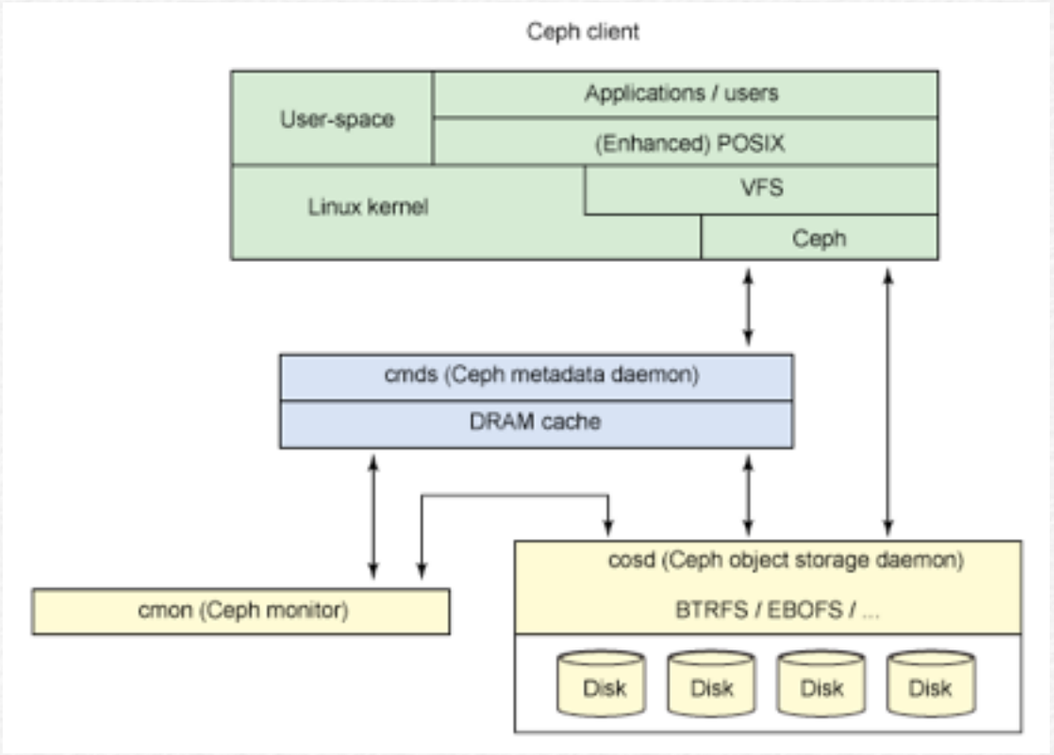
图2. Ceph 生态系统的简化分层视图



Ceph 组件

了解了 Ceph 的概念构图之后，你可以继续深入了解 Ceph 生态系统内实现的主要组件。Ceph 和传统文件系统一个关键的不同是，智能并非集中于文件系统本身，而是分布在生态系统各处。

图 3 展示了一个简单的 Ceph 生态系统。Ceph 客户端是 Ceph 文件系统的使用者。Ceph 元数据后台服务程序提供了元数据服务，而 Ceph 对象存储后台服务程序提供了实际的存储（数据及元数据）。最后， Ceph 监控提供了集群的管理。注意可以存在多个 Ceph 客户端，多个对象存储端点，许多元数据服务器（取决于文件系统的能力），和至少一对冗余监控。这样的话，这个文件系统是怎样实现分布的呢？图 3. 简单的 Ceph 生态系统



Ceph 客户端

因为 Linux 为文件系统提供了一个通用接口（通过虚拟文件系统交换器 [VFS]），Ceph 的用户视角是透明的。考虑到多个服务器组成存储系统的可能性（详见关于创建 Ceph 集群的资源 一节），管理员视角当然会不同。从使用者的视角看，他们可以使用海量存储系统，并不知晓下层的元数据服务器、监控、聚合成大规模存储池的各个对象存储设备。使用者仅仅看到一个挂载点，从那里执行标准的文件 I/O 操作。

Ceph 文件系统 — 或者至少是客户端接口 — 在 Linux 内核中实现。注意绝大多数文件系统，所有的控制和智能都实现在内核的文件系统源代码本身中。但对 Ceph 而言，文件系统的智能分布在各节点上，它简化了客户端接口，但也同时提供给 Ceph 应对大规模（甚至动态的）数据的能力。

并非是依赖于分配链（用来映射磁盘块到指定文件的元数据），Ceph 使用了一个有趣的选择。一个文件从 Linux 的角度看，被赋予了一个来自元数据服务器的 inode 号（I-NO），它是文件的唯一识别符。之后这个文件被刻进到多个对象中（基于文件大小）。使用 INO 和 对象号 （ONO），每个对象被赋予了一个对象 ID（OID）。通过使用 OID 上的一个简单哈希，每个对象被分配到 一个放置组中。这个放置组（通过 PGID 来识别）是对象的概念上的容器。最后，放置组到对象存储设备的映射是一种使用了称为可扩展哈希受控复制（CRUSH）算法的伪随机映射。以这种方式，放置 组（及副

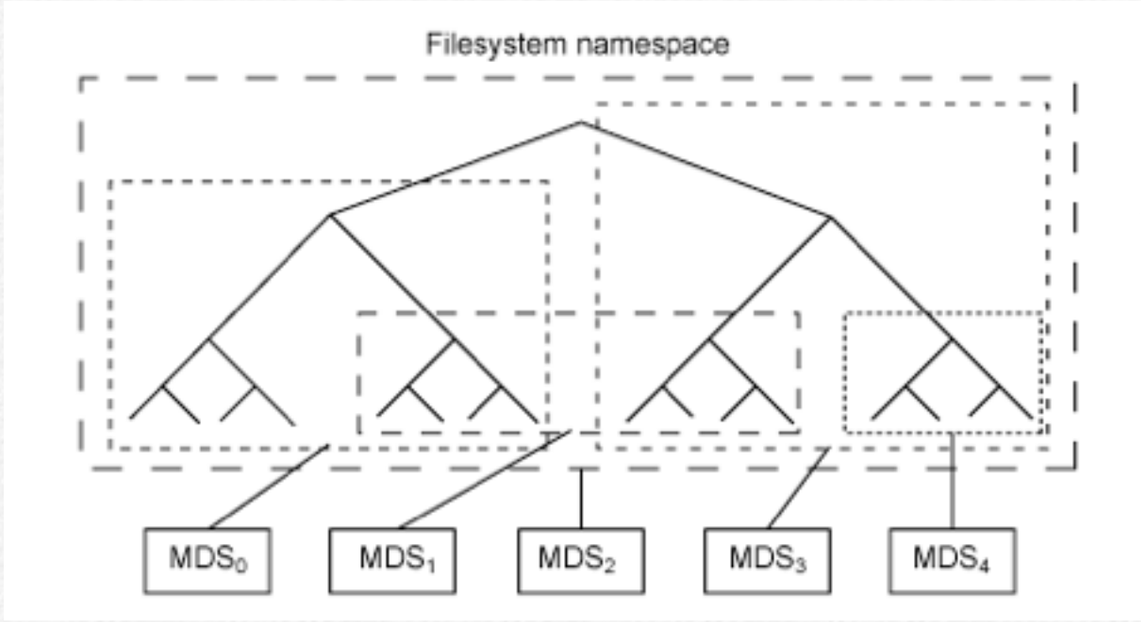
本）映射到存储设备不依赖于任何元数据，而是依赖于伪随机映射函数。这种行为是理想的，因为它最小化了存储的负荷，以及简化了数据的分布和查询。

最后一个分配组件是集群映射。集群映射是代表着存储集群的设备的有效表示。通过 PGID 和 集群映射，你可以定位任何对象。

Ceph 元数据服务器

元数据服务器的任务 (命令) 是管理文件系统的命名空间。尽管不管是数据还是元数据都存储在对象存储集群中，它们都被分别管理，以支持可扩展性。事实上，元数据在元数据服务器的集群间进一步的分割，这些元数据服务器能够自适应的复制和散布命名空间，从而避免热点。如图 4所示，元数据服务器管理部分命名空间，并能部分重叠（出于冗余及性能的考虑）。从元数据服务器到命名空间的映射，在 Ceph 中通过动态子树分区的方式来执行，它允许 Ceph 在保留本地性能的同时可以适应变化的工作负荷（在元数据服务器间迁移命名空间）。

图 4. Ceph 元数据服务器命名空间的分区



但因为每个元数据服务器就客户端的个数而言只简单的管理命名空间，它主要的应用程序是智能元数据缓冲（因为实际的元数据最终存储在对象存储集群 中）。写入的元数据被缓冲在短期的日志中，它最终会被推送到物理存储里。这种行为允许元数据服务器可以立即对客户端提供最新的元数据服务（这在元数据操作中很常见）。日志对故障恢复也很有用：如果元数据服务器失效，它的日志能被重播，以确保元数据安全的保存到磁盘上。

元数据服务器管理 inode 空间，进行文件名到元数据的转换。元数据服务器转换文件名到 inode 节点，文件大小，以及分割 Ceph 客户端用于文件 I/O 的数据（布局）。

Ceph 监控

Ceph 包括了实现集群映射管理功能的监控，但一些容错管理的要素实现于对象存储本身。当对象存储设备失效，或者新的设备被添加，监控检测并维持一个有效的集群映 射。这个功能以分布式的方式执行，在这里，更新映射会与当前的流量状况进行沟通。Ceph 使用了 Paxos，它是分布式一致性的一系列算法。

Ceph 对象存储

与传统对象存储相似，Ceph 存储节点不仅包含了存储，也包含了智能。传统的驱动器是的简单目标端，仅仅响应发起端的命令。而对象存储设备是智能设备，既可以作为目标端，也可以作为发起端，从而支持通讯以及其它对象存储设备的协同。

从存储的角度看，Ceph对象存储提供了从“对象”到“块”的映射（传统上这个事情在客户端的文件系统层来做）。这个行为允许本地实体对如何来存储对象做出最好的选择。早期的Ceph版本使用了一个在本地文件系统上的自定义的底层文件系统（EBOFS）。这个系统实现了一个非标准接口的底层存储来提供对象语义和其他特征（如对磁盘提交的异步通知）。现在可以在存储节点上使用B-tree文件系统（BTRFS），它已经实现了一些必要的功能（如嵌入式完整性）。因为Ceph客户端使用CRUSH，而不知道磁盘上文件的块映射信息， 所以底层的存储设备可以安全的管理从对象到块的映射。 这使得，当设备被发现失效后，依然节点复制数据。失效恢复的分布式机制也提升了存储系统的扩展性，因为失效检测和恢复在整个生态系统得到分布。Ceph中称之为RADOS（见图3）。

其它有趣的特性

似乎是文件系统的动态自适应的天性不足，Ceph 也实现了一些对使用者可见的有趣特性。例如，用户可以在 Ceph 的任何子目录下（包括其全部内容），创建快照。也可以在任何子目录级进行文件和容量记账，它可以报告存储大小和任何指定子目录的文件数（及其所有的嵌套内 容）。

Ceph 现状与未来

尽管 Ceph 现在集成到 Linux 主流内核中，它被准确的描述为实验性的。这种状态的文件系统有助于评估，但还没有为生产环境做好准备。但考虑到 Ceph 被 Linux 内核所采用，以及发起人持续开发的动力，它应该在不久后就可以用于解决你的大规模存储的需要。

其它分布式文件系统

Ceph 在分布式文件系统领域并不是独一无二的，但它在管理大规模存储的生态系统的方式上是独特的。其它的分布式文件系统，如包括 Google 文件系统（GFS），通用并行文件系统（GPFS），和 Lustre，仅举几例。随着海量级别存储的独特挑战的引入，Ceph 背后的理念看起来像是为分布式文件系统提供了一种有趣的未来。

展望

Ceph 不仅仅是文件系统，更是一个带有企业级特性的对象存储的生态系统。在 资源 一节中，你会发现关于如何建立一个简单的 Ceph 集群（包括元数据服务器、对象服务器、监控）的信息。Ceph 在分布式存储方面填补了空白，看到这个开源软件在未来如何演进也将是很有趣的。

原文链接: http://www.oschina.net/translate/ceph

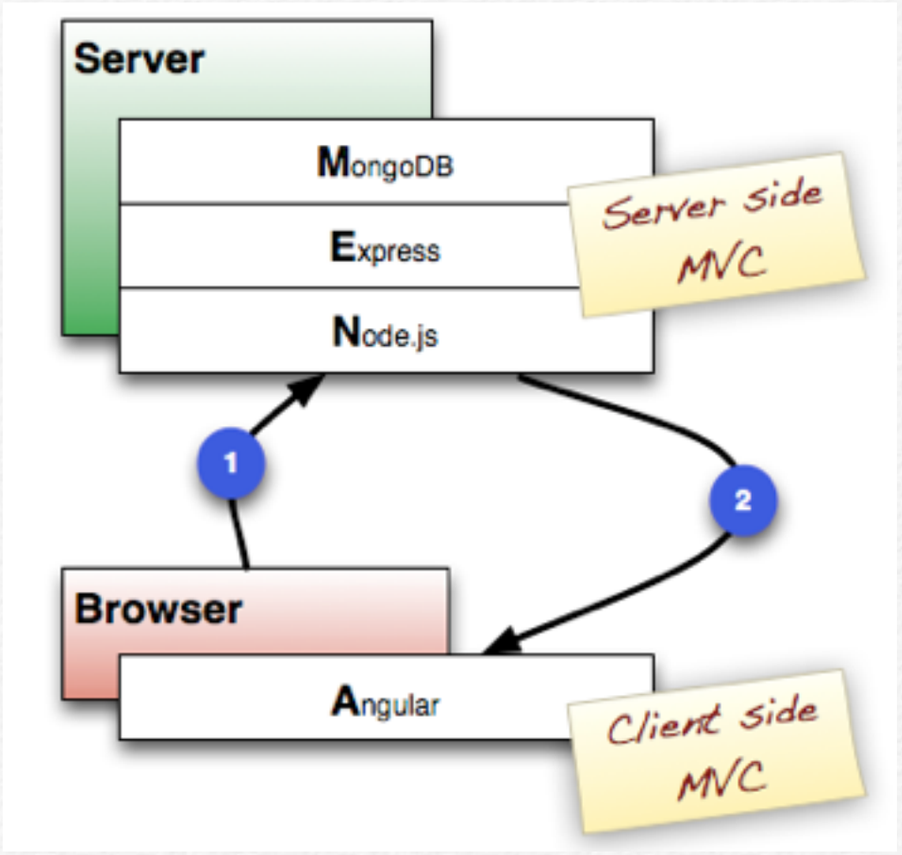
翻译： yxrykds, WangZhiping

译文链接：Ceph: A Linux petabyte-scale distributed file system

全 Javascript 的 Web 开发架构：MEAN

引言

最近在Angular社区的原型开发者间，一种全Javascript的开发架构MEAN正突然流行起来。其首字母分别代表的是：(M)ongoDB —— NoSQL的文档数据库，使用JSON风格来存储数据，甚至也是使用JS来进行sql查询；(E)xpress——基于Node的Web开发框架； (A)agular——JS的前端开发框架，提供了声明式的双向数据绑定；(N)ode——基于V8的运行时环境（JS语言开发），可以构建快速响应、可 扩展的网络应用。



MEAN架构的示意图

MEAN的支持者宣称，如果整个开发栈均能使用JS，必然会大大地提高效率，这一点毋庸置疑是一个很大的卖点。这样一来，开发人员（无论是前端还是后端）不仅能使用一致的数据模型，在其它方面也同样可以获得一致的编程体验。

例如，拿Mongo来说，你可以使用类JSON格式（BSON，二进制的JSON）来存储数据，然后在Express/Node中调用JSON查询语句，再将结果以JSON格式传给前端的Angular显示，这样，也自然使调试程序容易了很多。

注意: 事实上，在MEAN架构中，前端的Angular并不是必须的，你可以将它替换成其他的前端框架，如Backbone、Ember或者Polymer。

为何选择MongoDB?

如上所提，这个架构最重要的优势在于能使用单一语言，这也是其选择了Mongo的首要原因。 这里就不讨论noSQL的是与非了。一些人对MEAN架构的指责在于，MongoDB可以很好地胜任中小型的应用，但是对于大规模应用（百万级用户）来说 可能就捉襟见肘。我想说的是，这完全取决于你如何使用它。

SQL数据库本身是强类型的，因此可以在很大程度上保证某种层次的一致，从而确保许多类型的脏数据一开始就没办法进入数据库。而NoSQL则是弱类型的数 据库，这使得它在数据验证方面力不从心，而只能交给开发人员来实现，基于这样的特性，它尤其适合存储那些不规范的数据，特别是在原型开发阶段，此时数据模 型正在经历快速变化。

SQL和noSQL间的技术差别，归根结底是要在性能和稳定性间作出平衡。有些情况下，对数据的事务处理一旦设定后就不会轻易变化，那么此时使用 Mongo就非常合适；然而有时候也会涉及

更为复杂的事务处理，会牵扯到许多独立的业务逻辑，由于Mongo并没有提供一个简单的数据模型来支持一定级别 的原子操作，因此SQL在这个时候可以派上用场。

但无论如何，不论你是否选择MEAN中的M，你最终都需要根据自身的需求选择出合适的工具来做正确的事情，

为何是Express?

可以简单地把Express看成是一个在Node平台下搭建Web应用的工具集。在Node的基础上，它提供了许多简洁的接口来创建请求节点、处理cookie等，此外还提供了许多功能来帮助你搭建自己的服务器。总的来说，Express在以下几个方面有优势：

- 设置REST路由简单致极：
- app.get(/account/:id,function(req, res){/* req.params('id') is available */});
- 支持Jade或Mustache等模板引擎
- 自动HTTP头处理:
- app.get('/',function(req,res){ res.json({object:'foo'}); });
- 支持Connect中间件，可以插入额外请求或响应处理，如用户验证
- 提供辅助函数解析POST请求
- 防范XSS
- 优雅的错误处理

如何快速上手MEAN

如果想要很快上手MEAN，那么mean.io是一个很不错的选择。该项目旨在解决MEAN架构中一些常见的集成问题, 目前维护得很好，文档也很清楚，而且可以很方便地自行添加第三方库，还能和Yeoman配合使用（通过 generator-mean by James Cryer）。

在进一步阅读前，先假定我们同意以下观点：（a）Mongo至少非常适合于全Javascript堆栈的原型设计；（b） 承认即使像Angular般如日中天，终有一天也会被其他的一些JS框架给取代，只要它们能帮助我们快速方便地将这个架构快速搭建起来。

接下来就要介绍Yeoman 了，它由3个你所熟知的工具构成：

- Grunt：用于生成，预览和自动化测试你的项目，这要特别感谢由Yeoman和grin t-contrib团队创建的许多grunt tasks。
- Bower：前端的依赖管理工具，有了它你再也不需要手动下载和管理第三方JS库了。
- YO：快速生成一个新的应用，包括配置好你的Grunt任务以及你极有可能会用到的Bower依赖。

笔者在一年以前，曾和其他一些人创建过一个叫ExpressStack的项目，其想法很简单，就是要提供一些工具能够快速生成构建全JS的Web应用所需要的一切。但是，这个项目夭折了，尽管如此，许多类似的项目却如雨后春笋般涌现出来。

下面对这些项目作些介绍：

注意: 你可能需要装好Yeomam(npm install -g yo)和以下一些生成器（npm install -g <generator-name>）。

1. generator-angular-fullstack

这是一个AngularJS的生成器，集成了Express,可选MongoDB。主要功能如下：

- 支持客户端和服务端的Livereload。
- Express server集成了grunt tasks。
- 内建了方便的部署流程。
- 支持Jade。

可参考: http://tylerhenkel.com/creating-apps-with-angular-and-node-using-yeoman/

2. generator-meanstack

另一个MEAN架构的生成器，集成了grunt-express，功能如下：

- 在generator-angular的基础上，用express取代了Connect。
- 支持客户端和服务端的Livereload。
- 使用app_grunt.js文件来启动应用，而在app.js中定义路由。
- 目录结构基本沿袭了generator-angular的风格，只作了少许的改动。

可参考：https://github.com/Grievoushead/generator-express-angular

3. generator-mean-seed

集成了Mongo，Express，Angular，Yeoman，Karma和Protractor（作自动测试）。

4. generator-klei

和其他的很类似，不过使用的是Mongoose和Stylus，其他的一些功能包括：

- 其目录结构非常容易扩张（包含了一个TODO List的应用例子）
- 一个配置完整的Gruntfile，集合了livereload, linting, concatenation, minification等.
- 使用exctrl来自动挂载API。
- 使用了grunt-injector，从而无需手动修改Html的layout文件，就可以自动装载新添加的js和css。
- 使用Karma, Mocha和Chai来进行前端的单元测试。

5. ultimate-seed-generator

该生成器非常全面，添加了许多第三方的库，包括Passport用于用户登录，Browserify加载js。

- 集成了AngularUI, Barbeque（用于任务队列管理）和Boot-Strap
- 集成了Bower, Browserify，Express和Font Awesome
- 集成了Grunt, Handlebars, jQuery, JSHint和Karma/Mocha
- 支持LESS/LESSHat, Livereload和Lodash/Underscore

- 集成了Modernizr, MongoDB/Mongoose和Passport

该如何做出选择?

看了这么多的生成器，自然要问，我该选择哪一个呢？事实上，以上列表是有顺序的，依据的是其与最新版的Yeoman的兼容性以及维护的活跃度。

全Javascript的架构是否适合于产品级的应用呢?

不得不承认，如果开发堆栈的每一层都能使用JavaScript将会是一件很棒的事情（至少对于原型开发来说是这样），然而千万要注意，不要为了追求这一 目标，而把自己而束缚住了。尽管的确有越来越多的大规模应用都在采用类似的架构，如Walmart、LinkedIn，但并不意味着模仿他们就一定能成 功。

另一个需要注意的是，相对于其他的语言（如Ruby，Python或Java），在Node上搭建后端要困难得多。你可能要自己处理内存泄漏，避免在事件 循环中进行耗时运算，还要非常小心异常处理，如果这些处理不当就很有可能会导致整个应用服务器崩溃，但是这些问题在其他平台上却已经处理得很好了。然而，这并不是说，Node不能运用在生产环境下，当然可以，但要格外用心。

实话实说，想要“一揽子”为Web应用提供一个大而全的解决方案将非常困难，MEAN架构也肯定有其局限性。在前端与后端的设计模式、原则和风格中有大量的演化，如果你觉得PHP或Rails是更明智的选择，那就继续使用下去，否则的话，不妨试试MEAN，至少在当下值得一试。

原文链接：http://raohuaming.github.io/blog/2014/02/09/full-stack-javascript-with-mean-and-yeoman/

原译文：http://addyosmani.com/blog/full-stack-javascript-with-mean-and-yeoman/

译者：@饶华铭

干货来了！OSTC 2014 全部演讲幻灯大汇总！

2014年3月30日，由CSDN主办的“开源技术大会·2014” (简称OSTC 2014)在北京丽亭华苑酒店召开。以下是我们收集的大会演讲的PPT，欢迎在线预览。

主会场演讲

1. 蒋涛：OSTC 开源大会开幕演讲　　08:45~09:00

【演讲人介绍】CSDN 创始人、董事长

【PPT在线预览】 http://share.csdn.net/slides/2794

2. 陈磊：腾讯开源战略　　09:00~09:45

【演讲人介绍】腾讯社交网络事业群副总裁

【PPT在线预览】 http://share.csdn.net/slides/2793

3. Larry Wall：无心插柳——开源萌发在那个春天(Accidentally On Purpose: the Early Story of Open Source)　　09:45~10:30

【演讲人介绍】Perl 语言设计者，程序员、系统管理员、语言学家和作家。他设计了编程语言 Perl，并参与和引领开源软件运动。

【内容介绍】Larry 介绍了当年自己的开源故事，分享了对于“开源”的理解。

【PPT在线预览】 http://share.csdn.net/slides/2798

4. 严旭(清风)：豆瓣Code开源历程　　10:30~11:15

【演讲人介绍】严旭(清风)，豆瓣技术总监。长期活跃在 Python 社区，目前正在维护豆瓣 CODE 系统。

【内容介绍】豆瓣 CODE 两年历程回顾：Git 不是万能的，没有 Review 是万万不能的。

【PPT在线预览】 http://share.csdn.net/slides/2799

分会场一主题演讲

1. 王川(fantix)：Python 开源异步并发框架的未来　　13:30~14:00

【演讲者介绍】CTO @ FlowForge Games，80后开源爱好者，ArchLinux x32 主要 committer，Python程序员。2009年带领团队翻译Twisted文档，2012 年独立完成 gevent 到 Python 3 的迁移，2013 年创建关注于 asyncio 集成的 gevent3 项目。

【内容介绍】即将发布的 Python 3.4 将包含一个新的异步并发框架 asyncio，我将跟大家一起从零开始，了解各种 Python 开源异步并发框架的现状和互操作性，共同学习 asyncio 的简单用法，展望不久的将来 asyncio 能给 Python 的开源生态环境带来怎样的变化。

【PPT在线预览】http://share.csdn.net/slides/2850

2. 徐小东：像黑客一样使用 Linux 命令行　　14:00~14:30

【演讲人介绍】网名 Toy，LinuxTOY.org 资讯站点站长。Linux-Toy 及时报道 Linux 及开源领域最新资讯，分享精彩的应用程序、实用教程和技巧。

【内容介绍】通过与 GUI 相比较，介绍 Linux 命令行的优势和常用技巧。

【PPT在线预览】 http://share.csdn.net/slides/2801

3. 李道兵：重整工具箱——从开源软件到开放服务
14:30~15:00

【演讲人介绍】七牛云存储首席架构师，Debian 开发者，多个开源软件的作者和维护者，原盛大资深研究员。

【内容介绍】为大家介绍从开源软件到开放服务的变化，同时介绍作为一个架构师和开发者，需要去思考哪些问题。

【PPT在线预览】http://share.csdn.net/slides/2849

4. 王康：开源软件无线电项目及GNURadio介绍
15:00~15:30

【演讲者介绍】核物理专业出身，清华大学网管会(TUNA)首任会长。开源软件无线电主题博客 hackrf.net 的创办人，致力于在国内推广开源软件无线电。给 Linux Kernel 提交过 LTE USB Dongle 的驱动补丁，获得业余无线电台操作执照(A级)。

【内容介绍】GNURadio 开源软件无线电框架提供了全面的对各种通信系统常用的信号处理模块，并提供了完整的信号开发模块，并已经形成完善良好的开源生态环境。但由于长久以来没有价格平易近人的硬件外设方案，使得 GNURadio 的学习门槛较高。HackRF 的出现将使得开源软件无线电被大众普遍认识提供了可能。

【PPT在线预览】 http://share.csdn.net/slides/2805

5. 赵涛：Fedora 大使项目与 FUDCon　　15:30~16:00

【演讲人介绍】Fedora 中文社区负责人，清华大学开源协会负责人。

【内容介绍】此演讲会介绍 Fedora 大使项目的目标、任务、活动、管理运行及在中国的发展情况。特别地会对 FUDCon 做一介绍，鼓励公众关注、参与即将到来的 FUDCon APAC 在北京的活动。

【PPT在线预览】http://share.csdn.net/slides/2851

6. 陈阳(Emily Chen)：女性如何参与自由开源软件社区
16:00~16:30

【演讲人介绍】Oracle 研发中心高级工程师、GNOME.Asia 创始人；2010~2011 年度 GNOME 基金会董事，Mozilla 社区贡献者。

【内容介绍】主要介绍女性如何参与 FOSS 社区，并以 GNOME 社区为例进行介绍。

【PPT在线预览】 http://share.csdn.net/slides/2807

7. Thomas Yao：上海 Linux 用户组的运营与管理
16:30~17:00

【演讲人介绍】上海 Linux 用户组负责人、GitCafe.com 创始人；经常组织上海 Linux 用户组的活动，有着丰富的社区经验。

【内容介绍】上海 Linux 用户组是中国最早成立的本地开源社区，到 2014 年已经走过了 17 年，是国内最大也是最活跃的本地开源社区，本次演讲主要分享 SHLUG 的历史以及运营经验。

【PPT在线预览】无

分会场二演讲

1. Zoom.Quiet(周琦)：但行好事莫问前程　　13:30~14:00

【演讲人介绍】Python 中文社区创始人/
管理员。主持编撰《可爱的 Python》、翻译多本 Python 技术图书，参与多个开源项目的开发/运维/翻译/
宣传工作，有着丰富的社区经验。

【内容介绍】回顾中国技术社区的脉络，对比社区内外的文化差异。

【PPT在线预览】 http://share.csdn.net/slides/2809

2. 赵伟：基于开源软件构建腾讯大数据平台　　14:00~14:30

【演讲人介绍】腾讯数据平台部高级工程师，腾讯大数据平台分布式数据仓库 TDW 技术负责人，开源软件爱好者，是腾讯内部 Hive、PostgreSQL、ZFS 等开源技术入和推广者。目前负责 TDW 平台的稳定性、性能、易用性建设，同时负责 TDW 对外开源工作。

【内容介绍】介绍腾讯 TDW 项目。

【PPT在线预览】 http://share.csdn.net/slides/2810

3. 李学辉：CloudStack 社区与商业的融合　　14:30~15:00

【演讲人介绍】天云趋势高级技术经理，CloudStack 中国社区创办者，Apache CloudStack 提交者，负责 CloudStack 的技术顾问及 CloudStack 在中国的推广应用工作。

【内容介绍】阐述 CloudStack 社区的发展，并探讨如何实现基于 CloudStack 社区与商业的共荣。

【PPT在线预览】 http://share.csdn.net/slides/2811

4. 薛正华：跳出那些坑——OpenStack 线上系统实战集
15:00~15:30

【演讲人介绍】博士，用友公有云技术总监中国计算机学会高级会员、大数据专委会委员。原中科院副研究员，近十年一直专注于大规模计算系统、并行计算和大数据工作。

【内容介绍】介绍用友公有云虚拟机系统云机的架构及关键技术，重点和大家分享线上系统遇到的各类问题以及解决之道，实战价值极高。

【PPT在线预览】 http://share.csdn.net/slides/2812

5. 马全一：Docker 镜像的存储和 Docker Registry
15:30~16:00

【演讲人介绍】Docker 中文社区发起人，负责 Docker 的宣传和推广、组织关于 Docker 项目的线下活动。

【内容介绍】介绍关于 Docker Registry、Index 和 Registry 的区别，如何构建 Private Repository 等。

【PPT在线预览】 http://share.csdn.net/slides/2813

6. 王斌：互联网 Server 测试领域的开源机会　　16:00~16:30

【演讲人介绍】在网易主要从事互联网领域的开源工作，主要维护 tcpcopy、gryphon、nginx-hmux-module 等开源项目。

【内容介绍】互联网 Server 程序，测试必不可少，而真正可用的测试工具却很少。本文将从一个全新的角度来发掘这一领域的开源机会，希望激发更多的人来从事这方面的开源，从而为互联网测试提供更好的服务。

【PPT在线预览】 http://share.csdn.net/slides/2815

7. 王文睿：用 Node.js 和 HTML5 开发本地应用
16:30~17:00

【演讲人介绍】任职于 Intel 公司开源技术中心(OTC)，从事 Web 和浏览器技术开发工作。现在主要工作是开发和维护 Node-Webkit 项目。

【内容介绍】Node-Webkit 是一个支持用 HTML5 和 Node.js 技术编写本地应用程序的运行环境。它可以让 Web 开发者利用已有的知识，快速开发具有先进 UI 的本地应用。本次演讲介绍了 Node-Webkit 项目的功能、历史、实现技术以及应用场景。

【PPT在线预览】 http://share.csdn.net/slides/2816

8. 魏子钧：Touch the Web　　17:00~17:30

【演讲人介绍】前端开发工程师，独立游戏开发者，专注于研究 Web 技术在互动娱乐领域的应用。

【内容介绍】聊聊 Web 端触控技术的应用 并分享自己在这方面的一些经验和心得体会。

【PPT在线预览】 http://share.csdn.net/slides/2817

9. 张慧华：DWZ 富客户端框架与 HTML5　　17:30~18:00

【演讲人介绍】DWZ 富客户端(jUI)创始人之一，2005 年至今一直从事互联网相关开发工作。2005 年至今一直从事互联网相关开发工作。

【内容介绍】DWZ 富客户端框架开发实战，并介绍该框架的 HTML5 手机 App 跨平台解决方案。

【PPT在线预览】 http://share.csdn.net/slides/2818

分会场三主题演讲

1. 罗聪翼：Blender 开源电影　　13:30~14:00

【演讲人介绍】BlenderCN 中文社区负责人，Blender 基金会认证培训师，创立 BlenderCN 中文社区，被誉为“中国 Blender 第一人”。

【内容介绍】伴随开源软件 Blender 发展而诞生的另一种开源产物——开源电影已经发展了近 8 年，期间诞生了 4 部完整 CG 动画和真人电影，以及众多小型独立制作短片和实验项目，本次演讲将为大家简单介绍 Blender 和由其制作的开源电影。

【PPT在线预览】http://share.csdn.net/slides/2819

2. 张频：Firefox OS 释放移动的未来　　14:00~14:30

【演讲人介绍】火狐中国软件经理，2007 年毕业于清华大学，2010 年加入 Mozil-
la，主要从事火狐浏览器功能定制以及火狐操作系统开发。

【内容介绍】本次演讲主要包括 Firefox OS 介绍及最新进展情况。

【PPT在线预览】http://share.csdn.net/slides/2820

3. 尹晟宇：腾讯追风移动加速项目介绍　　14:30~15:00

【演讲人介绍】腾讯公司云平台部研发副总监。2005 年加入腾讯，先后负责腾讯支付、营销体系、分布式数据存储系统、大数据传输平台等的研发和管理工作。

【内容介绍】介绍腾讯追风移动加速项目

【PPT在线预览】http://share.csdn.net/slides/2821

4. 王兴博：讲述国内自主的游戏引擎 Genesis-3D 15:00~15:30

【演讲人介绍】Genesis-3D 产品总监

【内容介绍】Genesis-3D 是由搜狐畅游发布的一款 3D 游戏引擎。

【PPT在线预览】http://share.csdn.net/slides/2822

5. 蔡文智：Cocos2D-x 与游戏开发 15:30~16:00

【演讲人介绍】CocoStudio 技术经理。CocoStudio 与 Cocos2d-x 同为 Cocos 社区重要贡献者。

【内容介绍】CocosEngine 作为一套免费游戏开发引擎和工具，为广大开发者带来很大便利。本演讲主要介绍其发展和优势。

【PPT在线预览】http://share.csdn.net/slides/2823

6. 云风：使用 Lua 开发网络游戏 16:00~16:30

【演讲者介绍】简悦科技 CTO，“风魂”游戏引擎、Ejoy2D 开源游戏引擎作者

【内容介绍】介绍陌陌争霸是如何基于开源图形引擎 Ejoy2D 和服务器框架 Skynet 开发的。

【PPT在线预览】http://share.csdn.net/slides/2824

7. 李明：开源项目对游戏产业的贡献 16:30~17:00

【演讲人介绍】9 秒社团创始人，最大中文开源游戏社区 9miao.com 创始人，从事多年开源游戏开发最大中文开源游戏社区 9miao.com 创始人

【内容介绍】开源项目对游戏产业带来了数不尽的贡献，游戏产业的高速发展和开源是密不可分的。

【PPT在线预览】无

8. 王春生：以商业模式保障开源软件持续发展 17:00~17:30

【演讲者介绍】禅道项目创始人，人称“春哥”，曾任开源测试管理软件 bugfree 核心开发者和维护者；2009 年发起禅道项目管理软件至今，2013 年发起蝉知企业门户系统。

【内容介绍】和大家分享禅道软件是如何通过商业和开源的相互促进实现自身的发展和盈利的。

【PPT在线预览】http://share.csdn.net/slides/2825

9. 潘少宁：社区商业化探索 17:30~18:00

【演讲者介绍】原腾讯技术研发，公益技术社区“LAMP人”发起人

【内容介绍】公益技术社区如何可持续化运营，如何在保证用户体验的情况下探索商业化?

【PPT在线预览】http://share.csdn.net/slides/2826

闪电演讲：

- OSTC开源技术大会：李家智，Beetl-现代Java模板引擎
- OSTC开源技术大会：朱辉，HelloGCC小组和开源开发工具大会

大会直播专题：

- http://special.csdncms.csdn.net/OSTC2014/

大会相关报道：

- “开源技术大会•2014”成功举办 腾讯CSDN宣布战略合作
- 腾讯宣布与CSDN合作，共建中国最大开源社区
- 【OSTC分会场一】从技术细节到运营管理，畅谈开源世界的那些事
- 【OSTC分会场二】企业的开源实践：用之有道
- 【OSTC分会场三】从开源电影到开源游戏，异彩纷呈的开源盛宴

原文链接：http://code.csdn.net/news/2819106?

作者： CSDN CODE

Cocos2d-x 3.0带来了什么

触控科技近日发布了Cocos2d-x 3.0 RC版本，主打开放、原生、创新路线。作为全球三大游戏引擎之一，Cocos2d-x吸引了国内外一批优秀的开发者和游戏作品，包括《放开那三国》、《刀塔传奇》、《BadLand》、《勇者斗恶龙》、《Dragon City》、《Line Play》等。新版本的Cocos2d-x引擎在跨平台游戏开发、打造完整的工具链、包含完整开发流程方面，做了更进一步的尝试。为了更全面地了解Cocos2d-x 3.0的性能特性，《程序员》日前对触控科技副总裁、Cocos2d-x游戏引擎作者王哲进行了专访。

Cocos2d-x 3.0的性能优化

《程序员》：较之上一个版本，Cocos2d-x 3.0在哪些方面做了改进?

王哲：在Cocos2d-x 3.0中，我们着重改进了一些很基础的工作，从之前发布的alpha版本来看，大家最关心的主要围绕三点：性能、兼容性（尤其是Android手机的兼容 性）、CPU和内存消耗。在性能方面，我们优化最好的是自动技术，当一个游戏场景大出手机屏幕很多时，引擎会自动帮你把屏幕之外的东西给剔除掉，这使大场 景游戏的流畅度有2~8倍的提升。另一个是模拟合并，从技术支持的反馈来看，有50%的开发者不懂得如何使用这个功能，而在3.0中，该功能可以自己判断 是否开启，并且性能已非常接近于手动调试。在提升兼容性方面，我们测试了200款Android机型，较之2.2版本的85.71%，3.0版本的兼容性达到了90.71%。至于CPU和内存消耗，3.0比2.2版本的启动时间缩短了25%，CPU的平均占用率也降低了36%。

《程序员》：除了性能上的优化外，在具体的功能上你们做了哪些工作?

王哲：我们首先从新的渲染器着手。Cocos2d-x的核心结构就是场景结构树，它由一个根节点延伸出许多子节点（如图1所示），如 果根节点向右移动200个像素，它的子节点也会跟着旋转200个像素。这种设计的局限性在于，由于Draw节点和Visit节点非常紧密，要访问某个节点 就必须先画这个节点，由此造成批量渲染功能难以实现。而且如果想跨平台，就需要所有节点都支持新平台。我们建立了一个新的组件，它里面有一个渲染队列，每 个节点都会发布它的渲染指令，这样，我们就可以分解Draw命令、Visit访问节点，实现批量渲染，并且所有OpenGL都会集中在渲染器里，所以，实 现跨平台非常容易。其次是新的API，Cocos2d-x的代码中会看到一些Sprite C的用法，但它和C++是不同的语言，于是，我们将它改成了更符合C++程序员使用习惯的方式。例如，我们把2.0中使用的CCSprite改成了 Sprite，用不同的命名空间来为Sprite命名。

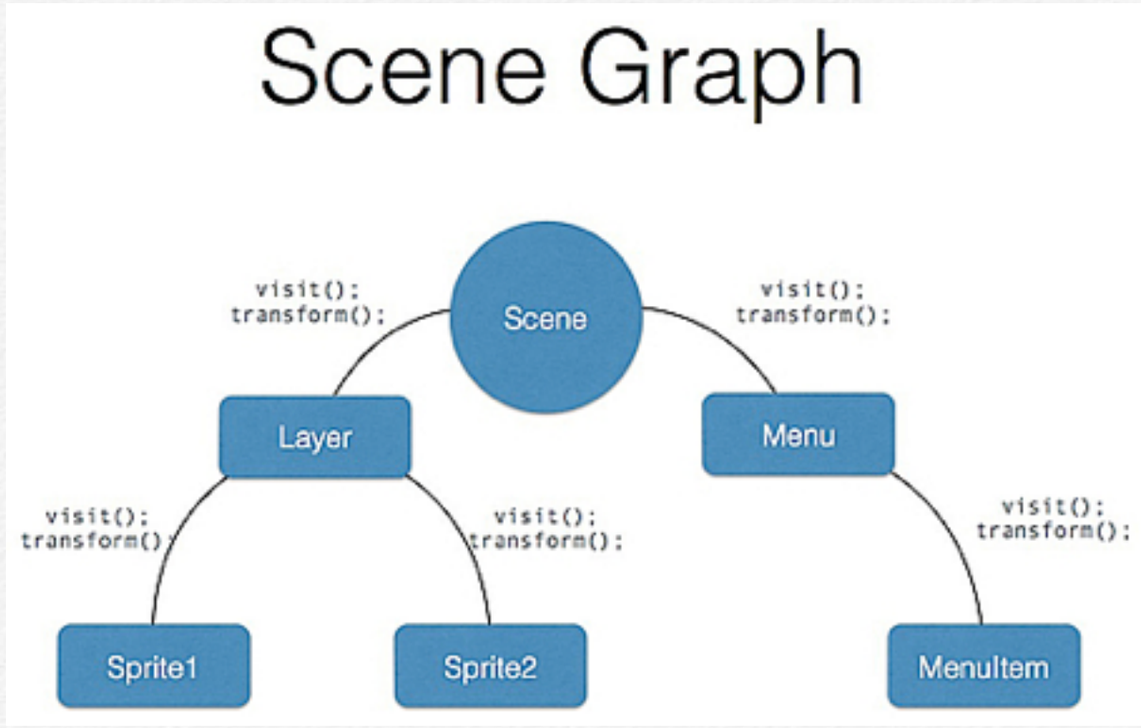


图1 Cocos2d-x的核心结构

《程序员》：有哪些修改和功能是来自开发者的反馈?

王哲：Critical Bug肯定是最先要修改的，此外还有平台不兼容，以及3.0中的XCode最新版本，有些开发者直接编译时挂掉了，这些问题肯定要优先解决。其他的还包括易用性，比如开发环境、Custom Draw以及Coding IDE等，这些功能都是大量开发者反馈需要的，所以我们就做出来了。

《程序员》：Cocos3d-x与Unity 3D引擎的主要区别是什么?

王哲：2D引擎发展到现在，表面上看其实都差不多，3D引擎也类似，最后拼的还是性能、效率，以及是否能用它做出比较炫的画面效果。 Cocos3d-x发展到今天，至少做出了像《捕鱼达人3》这样比较炫酷的游戏，有这样一款产品来验证它。可以说，在稳定性、高性能这些方面，我们已经做 得很好了，但在易用性上，说实话，差距还是挺大的，毕竟编辑器这些是Unity 3D的优势，但框架方面我们做得更好。

《程序员》：在技术实践中，有哪些团队对Cocos2d-x引擎的理解和操作堪称范例?

王哲：确实有一些，比如像《忘仙》和《君王2》。他们改掉了一些东西，如在TextureCache里增加警戒线，达到某个警戒线就触发一个回调，在回调里面释放掉一些资源，通过这种方式来保证它的内存不会爆。不过这对于一般休闲类游戏的作用不大，我没有添加这个功能主要是不想增加引 擎的复杂度。另一个是《刀塔传奇》，他们的技术人员很有意思，他们如果直接用Flash Dragon Bone格式导出，引擎可以直接解析。但他们觉得不爽，就自己写了一个Flash插件，把Flash里的骨骼动画导出到Cocos2d-x里解析出来。所 以这款游戏里的动作打斗场景都是他们自己做的一套。

增强培训，与开发者共赢

《程序员》：你们与开发者通过怎样的方式进行交流?

王哲：我们的Issue System是开放的，开发者认为哪里不好，可以直接提交任务。我们每天都会有人去查看，如果某个问题提出的人多的话，我们会马上分配开发团队去做，这是 一种很好的交流方式。此外，CocoaChina上有几万个帖子，我们也长期有专人维护，对开发者提出的问题进行统计，针对高频问题我们会马上修改。

《程序员》：除了外语教材外，你们在海外推广和对海外用户的技术支持方面还有哪些举措?

王哲：在推广方面我们做的不多，只会定期公布一些用Ccocos2d-x引擎开发的热门游戏的名单。在对海外用户的支持方面，目前，Cocos2d-x的教材已包含中、日、韩、英、西五种语言版本。在接触方面，应该还是跟日韩的开发者接触多一点（因为游戏风格和地理位置与中国都比 较接近），比如2013年11月在韩国举办的开发者沙龙等。此外，日本有一个自发Cocos2d-x学习社区叫“知友会”，他们会定期组织一些活动。 2013年12月我和林顺参加了他们组织的线下沙龙并做了分享。另外，他们还会帮我们把社区里的文档翻译成日语，以便更好地帮助本土开发者。

《程序员》：在你们与高校的合作中，Cocos2d-x课程的设置及合作模式是什么样的?

王哲：整个课程设置的核心围绕着Cocos系列工具，包括Cocos2d-x和CocoStudio等。但很多学生不仅想学习知识， 还非常关注行业动态、开发流程，甚至游戏运营等问题。基于此，我们共设置了9门课程，从游戏的开发概论、行业数据分析、游戏策划、美术，一直到游戏运营。 每个部分按单元来划分，每个单元32个课时。另外，我们还与学校合作，建立了实训基地，让

学生可以实际体验游戏开发的全过程。如果有人想创业，我们也会提 供技术和资金上的支持。

文章来源：《程序员》

作者：徐威龙

原文链接：http://www.csdn.net/article/2014-04-02/2819110

5种你未必知道的JavaScript和CSS交互的方法

随着浏览器不断的升级改进，CSS和JavaScript之间的界限越来越模糊。本来它们是负责着完全不同的功能，但最终，它们都属于网页前端技 术，它们需要相互密切的合作。我们的网页中都有.js文件和.css文件，但这并不意味着CSS和js是独立不能交互的。下面要讲的这五种 JavaScript和CSS共同合作的方法你也许未必知道!

用JavaScript获取伪元素(pseudo-element)属性

大家都知道如何通过一个元素的style属性获取它的CSS样式值，但能获取伪元素(pseudo-element)的属性值吗？可以的，使用JavaScript也可以访问页面中的伪元素。

```
// Get the color value of .element:before
var color = window.getComputedStyle(
     document.querySelector('.element'), ':before'
).getPropertyValue('color');

// Get the content value of .element:before
var content = window.getComputedStyle(
     document.querySelector('.element'), ':before'
).getPropertyValue('content');
```

看见了吗，我能访问伪元素里的content属性值。如果你想创建一个动态的，风格别致的网站，这是一种非常有用的技术!

classList API

很多的JavaScript工具库里都有addClass，removeClass和toggleClass等方法。为了对老式浏览器的兼容，这些类库采用的方法都是先搜索元素的className，追加和删除这个类，然后更新className。其实有一个新型的API提供了添加，删除和反转CSS类属性的方法，叫做classList:

```
myDiv.classList.add('myCssClass'); // Adds a class

myDiv.classList.remove('myCssClass'); // Removes a class

myDiv.classList.toggle('myCssClass'); // Toggles a class
```

大多数的浏览器里很早就实现了classListAPI，而且最终IE10里也实现了它。

直接对样式表进行添加和删除样式规则

我们都非常熟悉使用element.style.propertyName来修改样式，使用JavaScript能帮助我们做到这些，但你知道如何新增或修一个现有的CSS样式规则吗？其实非常的简单。

```
function addCSSRule(sheet, selector, rules, index) {
     if(sheet.insertRule) {
          sheet.insertRule(selector + "{" + rules + "}",
index);
     }
     else {
          sheet.addRule(selector, rules, index);
     }
}

// Use it!
addCSSRule(document.styleSheets[0], "header", "float:
left");
```

这种方法通常是用来创建一个新的样式规则，但如果你想修改一个现有的规则，也可以这样做。

加载CSS文件

延迟加载图片、JSON、脚本等是用来加快页面显示速度的好方法。我们可以使用curl.js等这样JavaScript加载器来延迟加载这些外部资源，可你知道CSS样式表也可以延迟加载吗，而且在加载成功后回调函数会给予通知。

```
curl(
    [
        "namespace/MyWidget",
        "css!namespace/resources/MyWidget.css"
    ],
    function(MyWidget) {
        // 你可以对MyWidget进行操作
        // 这里没有对这个css文件引用，因为不需要;
        // 我们只需要它已经加载到页面上了
    }
});
```

本网站使用的PrismJS语法高亮脚本就是延迟加载的。当所有的资源都加载后，回调函数就会触发，我可在回调函数里加载它。非常有用!

CSS鼠标指针事件

CSS鼠标指针事件pointer-events属性非常的有趣，它的功效非常像JavaScript，当你把这个属性设置为none时，它能有效的阻止禁止这个元素，你也许会说“这又如何？”，但事实上，它是禁止了这个元素上的任何JavaScript事件或回调函数!

```
.disabled { pointer-events: none; }
```

点击这个元素，你会发现任何你放置在这个元素上的监听器都不会触发任何事件。一个神奇的功能，真的——你不在需要为了防止某个事件会被触发而去检查某个css类是否存在。

就是这5给你也许还没有发现的CSS和JavaScript交互的方法。你还有新的发现吗？分享出来!

原译文：5 Ways that CSS and JavaScript Interact That You May Not Know About

译者：santiago

原文链接：http://www.webhek.com/ways-css-javascript-interact/

Mod 与 RequireJS/SeaJS 的那些事

本文的目的是为了能大让家更好的认识 Mod，之所以引入 RequireJS/SeaJS 的对比主要是应大家要求更清晰的对比应用场景，并不是为了比较出孰胜孰劣，RequireJS 和 SeaJS 都是模块化漫漫之路的先驱者，向他们致敬！

为工程化为生的Mod

模块化是一种处理复杂系统分解成为更好的可管理模块的方式，它可以把系统代码划分为一系列职责单一，高度解耦且可替换的模块，采用模块化可以让系统的可维护性更加简单易得。

JavaScript 并没有为开发者们提供以一种简洁、有条理地的方式来管理模块的方法。从出发点来看，Mod和 RequireJS/SeaJS 是一致的，为开发者提供一套 JavaScript 模块化开发方案，让 JavaScript 的模块化开发变得更简单自然。但是在实现的过程中却存在巨大着的差异。

Mod严格上来讲并不是一个独立的模块化框架，它是被设计用做前端工程化模块化方案的 JavaScript 支持，需要和自动化工具、后端框架配合来使用，目的在于希望给工程师提供一个类似 nodeJS 一样的开发体验，同时具备很好的线上性能。

RequireJS 和 SeaJS 的定位主要是 Web 浏览器端的模块加载器，依靠 JavaScript 运行时来支持模块定义、依赖分析和加载等功能。

类 CommonJS 的开发体验

RequireJS 遵守的是 AMD 规范，SeaJS 遵守的是 CMD 的规范。AMD/CMD 规范使用的是“异步模块定义”的方式，这种方式给开发带来了极大的不便，所有的同步代码都需要修改为异步的方式，我们是否可以在前端开发中使用“ CommonJS ”的方式，开发者可以使用自然、容易理解的模块定义和调用方式，不需要关注模块是否异步，不需要改变开发者的开发行为。答案当然是肯定的，Mod并不完全遵守 AMD/CMD 规范，也正是为了为开发者提供更简单自然的开发体验。

模块定义

Mod 使用 define 来定义一个模块：

```
define (id, factory)
```

factory 提供了 3 个参数：require, exports, module ，用于模块的引用和导出。

在平常开发中，我们无需关注模块定义，工具会自动对 JS 进行 define 包装处理：

JS 源码

```
//common/widget/menu/menu.js
var $ = require('common:widget/jquery/jquery.js');

exports.init = function() {
    $('.menu-ui ul li a').click(function(event) {
        var self = this;
        $('.menu-ui ul li
a.active').removeClass('active');
        $(self).addClass('active');
        event.preventDefault();
    });
};
```

编译后代码

```
define('common:widget/menu/menu.js', function(require, ex-
ports, module){
    var $ = require('common:widget/jquery/jquery.js');
    exports.init = function() {
        $('.menu-ui ul li a').click(function(event) {
            var self = this;
            $('.menu-ui ul li
a.active').removeClass('active');
            $(self).addClass('active');
            event.preventDefault();
        });
    };
});
```

模块调用

Mod会在模块初始化之前自动加载相关依赖。因此当我们需要一个模块时，只需提供一个模块名即可获取：

require (id)
因为所需的模块都已预先加载，因此 require 可以立即(同步)返回该模块引用。无论在页面的 script 还是模块内部，工程师都可以放心通过 require 来加载模块，不需要考虑何时该使用同步接口何时调用异步接口。

避免模块化引来的性能问题

RequireJS/SeaJS 通过过 JavaScript 运行时来支持“匿名闭包”、“依赖分析”和“模块加载”等功能，“依赖分析”需要在 JavaScript 运行时通过正则匹配到模块的依赖关系，然后顺着依赖链（也就是顺着模块声明的依赖层层进入，直到没有依赖为止）把所有需要加载的模块按顺序一一加载完毕， 当模块很多、依赖关系复杂的情况下会严重影响页面性能。Mod通过以下设计避免了如上问题：

- 通过工具自动添加 define 闭包，线上不需要支持匿名闭包
- 通过工具自动处理依赖，线上不需要动态处理依赖
- 通过后端模板自动插入 script，线上不需要通过前端框架进行模块加载

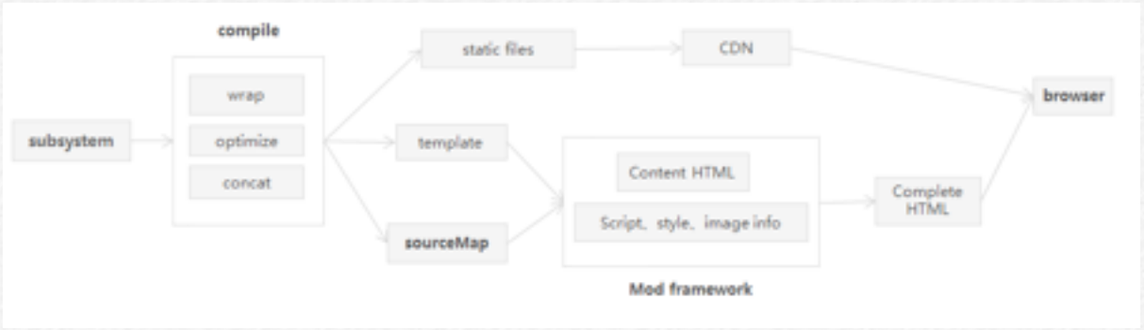
通过以上设计，Mod极其精简，整个文件只有 100 多行，相比下 RequireJS 有 2000 多行，SeaJS 有将近 1000 行。

避免模块化为打包部署带来的极大不便

通过 RequireJS/SeaJS 进行模块化开发后，合并静态资源(打包)将变得十分不方便和晦涩难懂，每个文件里只能有一个模块，无论是“ combo 插件”还是“ flush 插件”，都需要我们修改模块化调用的代码，这无疑是雪上加霜，工程师不仅需要在开发的时候关注模块定义，在调用的时候还需要关注在一个请求里面加载哪些模 块比较合适，模块化的初衷是为了提高开发效率、降低维护成本，但我们发现这样的模块化方案实际上并没有降低维护成本，某种程度上来说使得整个项目更加复杂 了。而使用 Mod，工程师只需要在配置文件配置合并策略即可，并不需要关注其他细节，Mod会自动处理好依赖以及合并信息并在模块初始化之前将模块的静态资源以及所依赖的模块加载并准备好。

自适应的性能优化

整个Mod模块化流程如下：



通过自动化工具对模块进行编译处理，包括对对 JavaScript 模块添加闭包、记录每个静态资源的部署路径以及依赖关系并生成资源表(resource map),如下所示，

```
{
        "res": {
            "demo.js": {
                "uri": "/static/js/demo_33c5143.js",
                "type": "js",
                "deps": [
                    "demo.css"
                ],
                "pkg": "p0"
            },
            "index.html": {
                "uri": "/index.html",
                "type": "html",
                "deps": [
                    "demo.js",
                    "demo.css"
                ]
            },
            "script.js": {
                "uri": "/static/js/script_32300bf.js",
                "type": "js",
                "pkg": "p0"
            }
        },
        "pkg": {
            "p0": {
                "uri": "/static/pkg/aio_5bb04ef.js",
                "type": "js",
                "has": [
                    "demo.js",
                    "script.js"
                ],
                "deps": [

                    "demo.css"
                ]
            }
        }
    }
```

所有被打包的资源会有一个 pkg 属性 指向该表中的资源，而这个资源，正是我们配置的打包策略。有了这些信息，我们可以通过 Mod 框架(Mod 和后端框架)来管理和控制模块的加载。Mod 的模块化可以十分灵活的适应各种性能优化场景，我们还可以通过监控模块的调用情况，自动生成最优的打包配置，让网站可以自适应优化。

总结

Mod提供的是一体化的模块化解决方案，更多的是从工程化、自动化的角度去考虑，RequireJS/SeaJS 更独立灵活。

相关阅读

- How to Develop With Widgets
- Concat Files And Manage Dependencies Automatically
- A Toolset For Production

作者：walter - F.I.S

原文链接：http://fex.baidu.com/blog/2014/04/fis-modjs-requirejs-seajs/

SegmentFault专访Face++ —— 世界领先的人脸识别云服务平台

Face++是 一个人脸识别云服务平台，通过它提供的开放服务，开发者可以快速地在自己的产品中集成面部识别功能。Face++团队专注于研发世界最好的人脸检测、识 别、分析和重建技术，通过融合机器视觉、机器学习、大数据挖掘及3D图形学技术，致力于将最新、性能最好、使用最方便的人脸技术提供给广大开发者和用户。

Face++的技术目前处于世界领先水平， 最近半年摘得多项世界技术比赛的桂冠：

1. 人脸检测（在照片中精准定位人脸位置的算法），在世界公开评测集FDDB 排名第一
2. 人脸关键点检测（精准定位面部关键部门的位置），在世界公开评测比赛300-W排名第一。
3. 在最重要的互联网图片人脸识别(Face recognition) 比赛LFW中，Face++团队更是力压Facebook人脸团队 (前face.com团队)，获得世界第一。在极难识别的互联网新闻图片上，获得了97.3%的准确率。

Face++为开发者提供了云端API、离线SDK，开发者可以方便地在自己的应用中集成人脸识别功能。

Segment-Fault访谈了Face++团队希望能帮助开发者更好地了解Face++。

SegmentFault: Face++能够应对复杂的光照并支持多种人脸姿态，能简单说一下背后采用的技术么?

Face++: Face++已经能够较为稳定地应对复杂的光照和人脸姿态，但过于极端的光照和姿态还是十分困难。Face++的全线技术都更新为大数据深度学习算法，通过收集的海量人脸照片，新型的神经网络算法可以通过大数据训练形成性能更好的人脸检测，人脸分析，和人脸识别技术。

SegmentFault: Face++大规模人脸搜索技术的快速高效主要得益于单张人脸特征长度小？除此以外有用其他技术优化搜索么？

Face++: 首先，Face++通过自己特有的face representation算法可以将每张人脸压缩到几个字节的长度。同时，我们在搜索建表，最近邻查找等技术也有自己独特优化的算法。

SegmentFault: Face++的人脸验证可以有效分辨活体和照片，不过绕过人脸验证的另一种方法是使用事先录制的用户视频。Face++是如何应对这种情况的？

Face++: 现在的Face++可以通过眨眼，表情变化等细微面部特征来分辨活体和照片。但和其他生物特征验证方式一样，Face++的活体技术短期内还无法应对如视频攻击这样的情况。

SegmentFault: Face++在ICCV2013上宣讲的Coarse-to-fine CNN技术，其中内部关键点检测之后还会有两级优化，但是比内部关键点识别难度还要大的轮廓关键点却没有后续的优化。这是为了减少步骤加快识别速度还是有其他的原因？

Face++: 我们其实试过内部和外部的多级优化，外部点因为比较不稳定，且ground-truth标准较难统一，所以多级优化后效果不明显，内部多级优化提升明显。

SegmentFault: Face++ 目前发布的两篇论文，关键点检测和人脸特征表示都是基于DCNN的技术。深度学习是人脸识别技术的发展方向么？

Face++: 未来深度学习一定是必须采用的方向。在我们看来深度学习和传统视觉方法有很强互补性，未来Face++也一定是两条路都要走。

SegmentFault: 百度余凯合著的深度学习论文得过ICML的Best Paper Runner-up Award，百度也成立了IDL（深度学习研究院），Google、Facebook也收购了一些人脸识别的公司。Face++ 目前具备世界领先的人脸识别技术，是怎么做到的？在Face++看来，在技术研发方面，创业公司相比巨头有哪些优势和劣势？

Face++: 百度，Google, 和Facebook都在深度学习上做的十分出色。在我们看来，大的技术革新最终靠的是人才。比如深度学习，大家都是初期的探索者，谁能更专注的进行更快速的技术革新，谁就会最终胜出。Face++在人才和技术专注度上都较为有信心。

SegmentFault: Face++的服务平台使用的是自建服务器还是第三方云服务？

Face++: 我们也是自建服务器和第三方云服务结合。

SegmentFault: Face++的算法研发用到了C++、Matlab和Python？能比较下Matlab和Python在视觉算法研究方面的优势和不足么？

Face++: 优势是比较糙快猛，方便验证idea；落实到系统和实用级别，可能还是要采用C++。

SegmentFault: Face++为iOS和Android提供了离线人脸检测，这和Face++的在线服务在功能上、识别准确率上有哪些差距？

Face++: 我们的离线人脸检测，可以理解为是一个轻量级版本，可以比较好的检出near-frontal的脸，能解决一些最基本需求。其他的功能还是线上服务性能更好，很快我们还会推出professional版本，以发布一些我们最新的技术，不过会限调用量。

SegmentFault: 网站介绍中说Face++的技术可以实时跟踪人脸，但是API中似乎没有找到相应的功能。以后会推出针对视频的人脸识别的API么？

Face++: 因为视频处理需要较多的计算资源，所以Face++会等自己各方面实力更强后再推出。

SegmentFault: 不同应用对人脸识别的要求可能是不一样的。比如人脸验证应用出于安全考虑，要保证尽可能低的FPR，而照片标记应用可能更偏向方便，兼顾FNR和FPR。以后Face++会推出针对性的服务么？（例如，API提供模式参数？）

Face++: 会，Face++未来会推出更厚的人脸服务，大家拭目以待。

SegmentFault: Face++团队最初的项目是结合视觉识别技术的体感操控游戏，目前主要为开发者、企业提供人脸识别技术的平台服务。以后的业务重心会偏向于提供平台服务，还是研发自己的产品呢？

Face++: Face++会一直作为一个出色的技术提供者存在。我们会一直优化我们的人脸技术和服务，未来还有可能从人脸拓展到其他图像领域。自己的产品会等技术打磨完成后再推进。

SegmentFault: Face++ 目前有多少员工呢？其中做技术的大概有多少？

Face++: 大约有20个全职，20个实习生。技术人员超过80%。

SegmentFault: 和一般创业公司不同，除了产品开发以外，Face++还需要进行算法科研。Face++内部从事开发和科研的人员比例大概是什么样的？

Face++: 开发和科研的比例大约是2:8左右。有很多出色的intern。

SegmentFault: 能介绍一下Face++的工作氛围么？

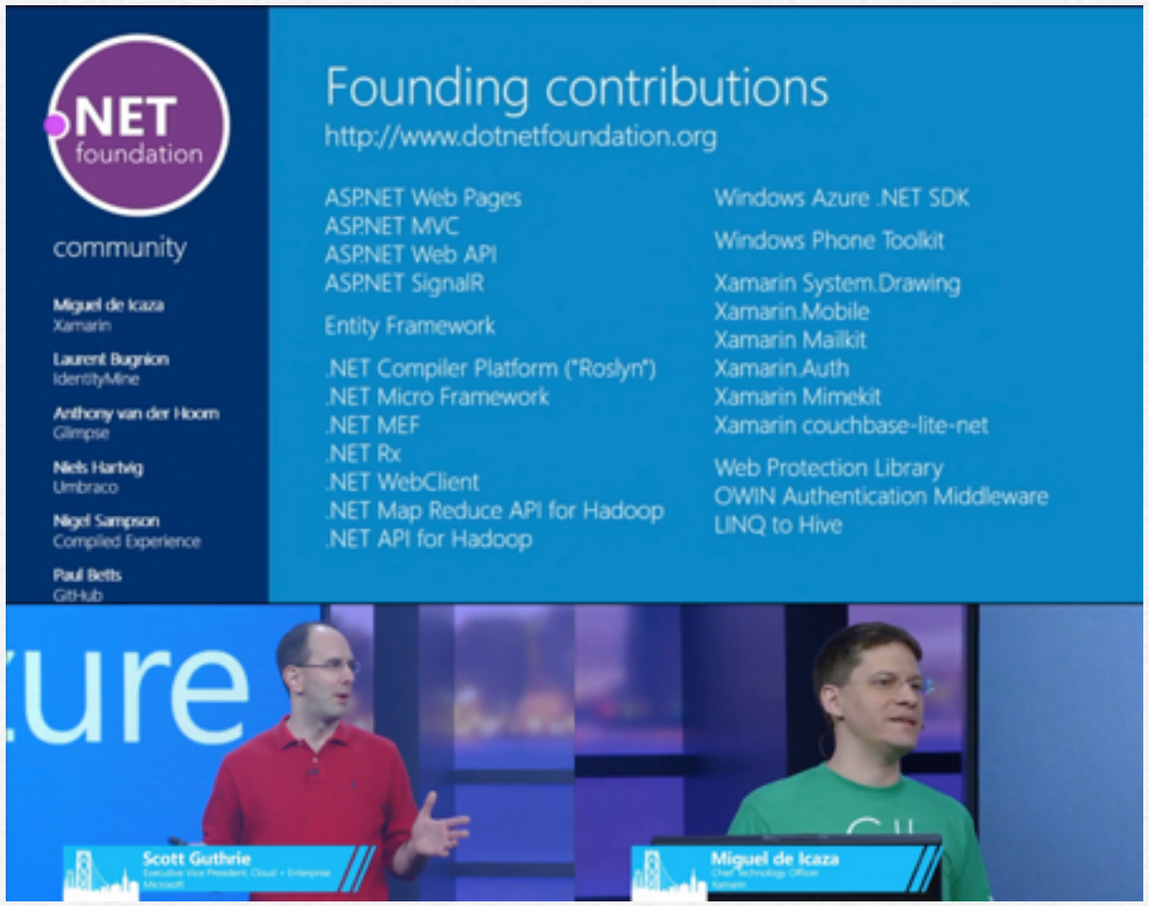
Face++: Face++是一个工程师氛围很浓的公司，理念是“Best is Endless. 追求极致.” 公司希望大家有硅谷创业的氛围，技术极客，快乐创业。

Face++休闲区

原文链接：http://segmentfault.com/a/1190000000452539

文章来源：SegmentFault

微软宣布成立.NET基金会全面支持开源项目



基金会初始董事包括 Mono 项目和 Xamarin 的老大 Miguel de Icaza，微软 .NET 团队代表和微软开放技术公司（这是微软专门为开源和开放技术、标准化成立的独立公司）代表。

首批 24 个项目包括 C# 编译器 Roslyn、ASP.NET 的多个已开源项目，还有 Xamarin 的几个项目。

Cloud and Enterprise 执行副总裁 Scott Guthrie 先生表示，今天宣布的决定是 .NET 基金会广泛倡议的一部分，公司将来开源更多的项目，现已启动 24 个 .NET 开源项目，大部分都是经 Apache 2.0 许可发布的。

当前项目列表包括:

.NET API for Hadoop WebClient
.NET Compiler Platform ("Roslyn")
.NET Map Reduce API for Hadoop
.NET Micro Framework
ASP.NET MVC
ASP.NET SignalR
ASP.NET Web API
ASP.NET Web Pages
Composition (MEF2)
Entity Framework
Linq to Hive
MEF (Managed Extensibility Framework)
OWIN Authentication Middleware
Rx (Reactive Extensions)
Web Protection Library
Windows Azure .NET SDK
Windows Phone Toolkit
WnsRecipe
Couchbase for .NET
Mailkit
Mimekit
Xamarin.Auth
Xamarin.Mobile
System.Drawing
Welcome to the .NET Foundation

基金会的官方网站：www.dotnetfoundation.org

原文链接：http://www.oschina.net/news/50437/dotnetfoundation

文章来源：开源中国社区 [http://www.oschina.net]

Linus Torvalds封杀了一名Red Hat的内核开发者

Linus Torvalds在内核开发者邮件列表上宣布封杀一名Red Hat的内核开发者，理由是 不想为他的代码“擦屁股”。Kay Sievers是一位知名的内核程序员，是systemd的关键开发者之一，然而他有一个不好的习惯是推卸责任，他递交的代码多次导致了内核问题，而他自 己并不愿意去修正自己代码的问题，Torvalds表示在这种习惯改变前他不会向内核合并任何来自Kay的代码，他说“再说一遍，如果你导致了问题，你就 应该去修正，而不是我想怎么做就怎么做”。

原文链接：http://www.solidot.org/story?sid=39028